

*DB2 9 Database
Administration Workshop for
Windows*

(Course code CF23)

Student Notebook

ERC 8.3

IBM certified course material

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX	AIX 5L	Approach
CICS	ClearCase	DB2
DB2 Connect	DB2 Universal Database	Distributed Relational Database Architecture
DRDA	Encina	Enterprise Storage Server
Informix	iSeries	Lotus
MQSeries	MVS	MVS/ESA
OS/2	OS/390	OS/400
pSeries	pureXML	QMF
QBIC	QuickPlace	RAMAC
S/390	Tivoli	z/OS
zSeries	1-2-3	

Alerts® is a registered trademark of Alphablox Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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

April 2007 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. 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 result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

© Copyright International Business Machines Corporation 1999, 2007. All rights reserved.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks	xiii
Course description	xv
Agenda	xix
Unit 1. Overview of DB2 9 on Linux, UNIX and Windows	1-1
Unit objectives	1-3
1.1 Overview of DB2	1-5
DB2 products	1-6
DB2 — The scalable DB	1-7
Accessing DB2	1-9
DB2 product components	1-12
DRDA Application Requester (AR) support	1-14
DRDA Application Server (AS) support	1-16
Installation prerequisites	1-17
Installation steps	1-19
For more information	1-21
Unit summary	1-22
Unit 2. Command Line Processor (CLP) and GUI usage	2-1
Unit objectives	2-2
2.1 Command Line Processor (CLP)	2-3
CLP Command Line Processor	2-4
CLP syntax	2-5
Online reference	2-6
Using the CLP	2-7
QUIT/TERMINATE/CONNECT RESET differences	2-9
CLP command options	2-11
Modify CLP options	2-13
Input file — no operating system commands	2-15
Input file — operating system commands	2-16
2.2 Getting started with DB2 GUI tools	2-17
Roadmap to the GUI tools	2-18
Control Center	2-19
Command Editor	2-21
SQL Assist window	2-23
Administration tools	2-25
Using the wizards	2-27
Health Center	2-30
Journal	2-32
Configuration Assistant (1 of 2)	2-34
Configuration Assistant (2 of 2)	2-36
Tool settings	2-38

2.3	Overview of the Administration Server	2-41
	DB2 Administration Server	2-42
	Using the Administration Server	2-44
	Unit summary	2-46
	Unit 3. The DB2 environment	3-1
	Unit objectives	3-2
	What is an Instance?	3-3
	The Database Manager instance	3-5
	Create and drop an instance	3-6
	db2icrt command	3-8
	db2idrop command	3-11
	Starting and stopping an instance	3-13
	Types of configuration	3-15
	Database Manager configuration	3-16
	Unit summary	3-18
	Unit 4. Creating databases and data placement	4-1
	Unit objectives	4-2
4.1	Create database overview	4-3
	Create database overview	4-4
	Steps for database creation	4-5
	Create Database Wizard	4-7
	Create database syntax (1 of 3)	4-9
	Create database syntax (2 of 3)	4-12
	Create database syntax (3 of 3)	4-15
	Automatic storage — database	4-18
	Automatic storage syntax — database	4-19
	Automatic storage — create database examples	4-20
	Automatic Storage provisioning — syntax	4-21
	Create bufferpool syntax	4-23
	Default table spaces and buffer pool	4-24
	System Catalog tables and views	4-25
4.2	Table spaces and containers	4-29
	DMS versus SMS table spaces	4-30
	Writing to containers	4-32
	Table space, container, extent, page	4-34
	DMS table space — minimum requirements	4-35
	Defining table spaces	4-36
	Create Table Space Wizard	4-37
	CREATE TABLESPACE syntax (1 of 2)	4-40
	CREATE TABLESPACE syntax (2 of 2)	4-43
	Automatic storage — table space examples	4-45
4.3	Gathering system information	4-47
	Get snapshot for tablespaces command	4-48
	Using the SNAPCONTAINER view	4-51
	Alter Table Space Wizard	4-52

ALTER TABLESPACE syntax	4-54
Placement considerations	4-57
4.4 Describe and modify the Database Configuration	4-59
Database Configuration	4-60
Database Manager Configuration	4-62
Database directories overview	4-63
Checkpoint	4-65
Unit summary	4-66
Unit 5. Creating database objects	5-1
Unit objectives	5-2
5.1 DB2 object hierarchy and physical directories and files	5-3
DB2 object hierarchy	5-4
Database physical directories and files	5-6
5.2 Creating objects	5-9
Create Schema	5-10
Set current schema	5-13
Create Table	5-15
Create Table statement	5-17
Table partitioning	5-20
Create View	5-22
Create View statement	5-23
Create Alias	5-25
Create Alias statement	5-26
Overview of Declared Temporary Tables	5-27
Create Index	5-29
Create Index statement	5-30
Overview of Referential Integrity	5-33
RI — Add a foreign key	5-36
Referential Integrity — Create Table statement	5-37
Unique Key considerations	5-39
Overview of Check Constraints: The need	5-41
Check Constraints — definition	5-42
Implementation of Check Constraints	5-44
Overview of triggers: The need	5-45
Implementation of triggers (1 of 2)	5-46
Implementation of triggers (2 of 2)	5-47
Create Trigger statement	5-48
Large objects: The need	5-51
5.3 Creating tables with XML columns	5-55
What is XML? (1 of 2)	5-56
What is XML? (2 of 2)	5-57
The XML document tree	5-58
PureXML datastore	5-60
Integration of XML and relational capabilities	5-62
Two ways to query XML data	5-64
For more information	5-67

Unit summary	5-70
Unit 6. Moving data	6-1
Unit objectives	6-2
Review INSERT statement	6-3
6.1 Export/Import utilities	6-5
Differences between IMPORT and LOAD	6-6
IMPORT/EXPORT overview	6-7
EXPORT command	6-9
EXPORT command syntax (basic)	6-10
EXPORT command example	6-13
IMPORT command	6-14
IMPORT command syntax (basic)	6-15
IMPORT command example	6-19
Import command... CREATE... IN TABLESPACE	6-21
6.2 LOAD utility	6-23
Four phases of Load	6-24
LOAD: Load phase	6-26
LOAD: Build phase	6-27
LOAD: Delete phase	6-28
LOAD: Index Copy phase	6-29
Load command	6-30
LOAD command syntax (basic)	6-32
LOAD scenario	6-35
Rules for creating Exception Table	6-37
Suggested methods for creating Exception Tables	6-39
Offline versus Online Load	6-40
Table states	6-42
Load Pending state: Recovering from LOAD failure	6-44
Backup Pending state — COPY options	6-46
Set Integrity Pending table state	6-48
SET INTEGRITY command overview	6-50
SET INTEGRITY command syntax ((basic))	6-51
Set Integrity Pending state	6-53
Meaningful steps for LOAD	6-55
Overview of the db2move tool	6-56
db2move tool syntax	6-57
db2move tool examples	6-60
Overview of the db2look tool	6-61
db2look tool syntax	6-63
db2look tool examples	6-66
Unit summary	6-69
Unit 7. Backup and recovery	7-1
Unit objectives	7-2
7.1 Introduction to backup and recovery	7-3
Backup and recovery	7-4

Types of failure	7-6
Database objects	7-7
Forms of backup	7-9
7.2 Methods of recovery and logging	7-11
Methods of recovery	7-12
Introduction to logging	7-14
Log file use during concurrent transactions	7-16
Circular logging	7-17
Archival logging	7-19
Mirrored logging	7-22
Log file information	7-23
Location of log files	7-24
Configure database logs — parameters	7-26
Configure database logs — GUI (Control Center)	7-32
Configure database logs — Wizard (CC)	7-33
Configure database logs — Command line	7-34
Recovery history file	7-35
7.3 Backup and restore utilities	7-39
BACKUP command	7-40
Syntax of the BACKUP command	7-42
BACKUP using the GUI interface	7-46
Online versus offline BACKUP	7-48
The backup files	7-50
RESTORE command	7-53
Syntax of the RESTORE command	7-55
RESTORE using the GUI interface	7-57
Backup/restore table space considerations	7-59
Roll forward pending	7-61
Syntax of the ROLLFORWARD command	7-63
ROLLFORWARD — GUI interface	7-67
ROLLFORWARD — how far?	7-69
Log file naming	7-71
Disaster recovery considerations	7-73
Questions to consider	7-75
Logging/backup requirements summary	7-76
Advanced recovery features	7-77
DB2 Advanced Recovery and HA Workshop	7-84
Checkpoint	7-87
Unit summary	7-88
Unit 8. Locking and concurrency	8-1
Unit objectives	8-2
8.1 Locking objects, strategies, and modes	8-3
Why are locks needed?	8-4
Classification of locks	8-6
Objects of locks	8-7
Locking strategies	8-8

Table lock modes	8-10
Row lock modes	8-13
Lock mode compatibility	8-15
8.2 Isolation levels	8-19
Introduction to isolation levels	8-20
Isolation levels	8-21
Repeatable Read versus Read Stability	8-23
Summary of isolation levels	8-25
SET CURRENT ISOLATION	8-26
8.3 Explicit versus implicit locking	8-29
Explicit versus implicit locking	8-30
LOCK TABLE statement	8-32
Lock conversion	8-34
Lock escalation	8-36
8.4 Lock timeouts and deadlock	8-39
Lock wait and timeout	8-40
SET CURRENT LOCK TIMEOUT	8-42
Deadlock causes and detection	8-45
Checkpoint	8-49
Unit summary	8-51
Unit 9. Problem determination	9-1
Unit objectives	9-2
9.1 Problem solving	9-3
How to solve a problem	9-4
Diagnostic data and data checklist	9-6
First Failure Data Capture (FFDC)	9-7
FFDC structure	9-11
Interpreting the Notification Log	9-13
Diagnostic Log Analysis Tool: db2diag	9-16
9.2 Database monitoring	9-23
Why use database monitors?	9-24
Ways to monitor the database	9-25
Snapshot Monitor — who and how?	9-27
Snapshot Monitor — when?	9-28
Snapshot Monitor — what?	9-29
Snapshot Monitor — CLP commands	9-30
Snapshot Monitor — taking snapshots	9-32
Event Monitor	9-34
Filtering Event Monitor output	9-36
Examining Event Monitor output	9-37
Activity Monitor	9-38
Health Monitor	9-40
Health indicator settings	9-42
Health Monitor — alerts view	9-45
Alarm details	9-47
Independent trace facility — db2trc	9-50

9.3 EXPLAIN	9-53
Explain tool	9-54
EXPLAIN facility tools	9-55
Visual explain access plan graph	9-57
9.4 Additional commands	9-59
Additional commands	9-60
LIST ACTIVE DATABASES command	9-62
LIST APPLICATIONS command	9-63
FORCE APPLICATION command	9-64
The db2set command	9-66
Retrieve stats from a running instance: db2pd	9-67
Performance tuning and monitoring workshop	9-72
Checkpoint	9-75
Unit summary	9-76
Unit 10. Application issues and performance	10-1
Unit objectives	10-3
10.1 Application alternatives	10-5
Application alternatives	10-6
10.2 Database engine access	10-9
Database engine access	10-10
Performance — dynamic versus static SQL	10-12
Database Manager APIs	10-14
10.3 Embedded SQL program preparation	10-19
Programming interfaces	10-20
Embedded SQL applications	10-21
Write the program	10-23
Program preparation steps	10-25
Associating load modules and packages	10-28
Precompiling versus binding	10-30
Adjusting qualifier and owner	10-31
Precompile command syntax (basic)	10-32
Bind command syntax (basic)	10-35
Package catalog views	10-37
Application process	10-39
Developer Workbench and SQL PL	10-40
10.4 Application performance	10-43
DB2 Optimizer	10-44
Optimization classes	10-46
Adjusting the QUERY OPTIMIZATION class	10-48
Design Advisor	10-50
Implicit rebinding	10-52
Explicit rebinding	10-54
Sorts — piped versus non-piped	10-56
Sorts — Sort passes	10-58
Concurrent application tuning introduction	10-59
Asynchronous Page Cleaner	10-61

10.5	Minimizing communication overhead	10-63
	Minimizing communication overhead	10-64
	Levels of data transmission	10-65
	Stored procedures	10-67
10.6	Utilities and tools	10-69
	Operational utilities	10-70
	Clustered and non-clustered indexes	10-72
	RUNSTATS command syntax (basic)	10-74
	Determine if table reorganization is required	10-77
	REORGCHK statistics	10-80
	Reorganize table / index command syntax	10-82
	Statistical Catalog views	10-85
	The db2look mimic mode	10-87
10.7	Additional learning opportunities	10-89
	Need more information? (1 of 3)	10-90
	More information (2 of 3)	10-92
	More information (3 of 3)	10-94
	Checkpoint	10-97
	Unit summary	10-100

Unit 11. Security	11-1	
	Unit objectives	11-2
11.1	Authentication versus authorization	11-3
	Authentication versus authorization	11-4
11.2	Access control — privileges	11-7
	DB2 access control hierarchy	11-8
	Database Authority summary	11-10
	Authorities and privileges	11-12
	Controlling use of schemas	11-16
	Protecting resources through programs	11-18
11.3	Explicit privileges	11-19
	Explicit authorization	11-20
	Grant statement — table / view privileges syntax	11-21
	Individual IDs, group IDs, and public	11-23
	Using group IDs	11-24
	Public privilege support for static SQL and views	11-26
	Privileges required for programming	11-28
	Dynamic rules and dynamic SQL	11-29
	DB2 BIND GRANT option	11-31
11.4	Implicit privileges	11-33
	Implicit privileges (1 of 2)	11-34
	Implicit privileges (2 of 2)	11-35
	Implicit privileges scenarios	11-37
	Grant / Revoke scenarios	11-38
11.5	Querying privileges/authorities	11-39
	Query who has privileges	11-40
	Query privileges granted to current ID	11-42

11.6 Label Based Access Control (LBAC)	11-43
Label Based Access Control (LBAC) overview	11-44
Security model extensions — SECADM Authority	11-46
Checkpoint	11-49
Unit summary	11-50
Appendix A. LOBs, UDFs, UDTs, and Relational Extenders	A-1
Appendix objectives	A-2
LOBs	A-3
Large objects — memory consideration	A-4
Large objects — manipulation	A-5
DB2 Relational Extenders	A-7
User-Defined distinct types and User-Defined functions	A-10
User-Defined Types — the need	A-11
User-Defined Types — definition	A-12
Create Distinct Type — GUI	A-14
User-Defined Functions — the need	A-15
User-Defined Function — definition	A-16
User-Defined Function — sourced	A-18
User-Defined Functions — fencing	A-20
Checkpoint	A-23
Appendix summary	A-24
Appendix B. Checkpoint solutions	B-1
Index	X-1

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX®	AIX 5L™	Approach®
CICS®	ClearCase®	DB2®
DB2 Connect™	DB2 Universal Database™	Distributed Relational Database Architecture™
DRDA®	Encina®	Enterprise Storage Server®
Informix®	iSeries™	Lotus®
MQSeries®	MVS™	MVS/ESA™
OS/2®	OS/390®	OS/400®
pSeries®	pureXML™	QMF™
QBIC®	QuickPlace®	RAMAC®
S/390®	Tivoli®	z/OS®
zSeries®	1-2-3®	

Alerts® is a registered trademark of Alphablox Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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

Course description

DB2 9 Database Administration Workshop for Windows

Duration: 4 days

Purpose

This course is designed to teach you how to perform database administration tasks using DB2 9. These tasks include creating DB2 instances, creating and populating databases, and using logical design to support concurrency and recovery requirements. New features, such as range partitioning, data row compression, and pureXML (native XML storage) will be introduced.

Audience

System administrators, database administrators, and technical personnel involved in planning, implementing, and maintaining DB2 databases.

Prerequisites

Before taking this course you should be able to:

- Use basic OS functions such as utilities, file permissions, hierarchical file system, commands, and editor
- State the functions of the Structured Query Language (SQL), and be able to construct DDL, DML, and authorization statements
- Discuss basic relational database concepts and objects such as tables, indexes, views, and joins

These skills can be developed by taking:

- OS Training
 - AIX 5L Basics
 - Linux Basics and Administration
 - Windows Systems Administration
 - Or by having equivalent HP-UX or Solaris administration experience
- DB2 SQL Workshop
- DB2 Fundamentals

Objectives

After completing this course, you should be able to:

- Administer a DB2 database system using commands and GUI tools
- Implement DB2 security
- Manage System Managed Storage (SMS) and Database Managed Storage (DMS) databases and apply data placement principles
- Implement a given logical database design using DB2 to support integrity and concurrency requirements
- List and describe the components of DB2
- Define a DB2 recovery strategy and perform the tasks necessary to support the strategy
- Describe the application development process with respect to DB2 considerations
- Use autonomic features of DB2

Contents

Overview of DB2 9 on Linux, UNIX and Windows

Command Line Processor (CLP) and GUI usage

The DB2 environment

Creating databases and data placement

Creating database objects

Moving data

Backup and recovery

Locking and concurrency

Problem determination

Application issues and performance

Security

Curriculum relationship

CF03 DB2 Family Fundamentals

CF53 DB2 Fundamentals

CF20 DB2 9 Database Administration Workshop for Linux

CF21 DB2 9 Database Administration Workshop for UNIX

CF24 DB2 Multi Partition DBA Workshop

CF28	Fast Path to DB2 9 for Experienced Relational DBAs
CF50	DB2 V6 Administrator Certification Crammer
CF10	DB2 Programming Fundamentals
CF11	DB2 Advanced Programming
CG11	DB2 for Linux, UNIX, and Windows Programming using Java
CF41	DB2 Performance Monitoring and Tuning Workshop
CF44	DB2 Multi Partitioned Performance Monitoring and Tuning Workshop for UNIX
CF46	DB2 for Linux, UNIX, and Windows Advanced Database Administration for Experts
CF15	Web Enabled DB2: Net.Data
CF60	DB2 Connect DRDA Implementation with TCP/IP
CF63	DB2 Connect to DB2 for OS/390 Problem Determination and Performance
CF49	DB2 Advanced Recovery and High Availability Workshop
CG07	DB2 UDB for UNIX, Windows and OS/2 V7 Transition
CG24	DB2 Multi Partition Environment for Single Partition DBAs
CF48	DB2 9 for Linux, UNIX, and Windows Quickstart for Experienced Relational DBAs
DW48	DB2 Federated Database with DB2 Information Integrator: Up and Running
DW49	DB2 Federated Database Performance, Tuning, and Advanced Topics Workshop

Agenda

The planned agenda follows. Here are the considerations:

- The first five units must be taught in the order specified:
 - Overview of DB2
 - Command Line Processor (CLP) and GUI usage
 - The DB2 environment
 - Creating databases and data placement
 - Create database objects
- Moving data
- Backup and recovery
- Problem determination
- Locking and concurrency
- Application issues and performance
- Security

Day 1

Welcome

Unit 1 - Overview of DB2 9 on Linux, UNIX and Windows

Lab 1 (Starting your lab environment)

Unit 2 - Command Line Processor (CLP) and GUI usage

Lab 2 (DB2 Customization)

Unit 3 - The DB2 environment

Lab 3 (DB2 environment)

Day 2

Unit 4 - Creating databases and data placement

Lab 4 (Creating databases and data placement)

Unit 5 - Creating database objects

Lab 5 (Create objects)

Unit 6 - Moving data

Day 3

Lab 6 (Moving data)

Unit 7 - Backup and recovery

Lab 7 (Backup and recovery)

Unit 8 - Locking and concurrency

Lab 8 (Investigating DB2 locking)

Day 4

- Unit 9 - Problem determination
- Lab 9 (Problem determination)
- Unit 10 - Application issues and performance
- Lab 10 (Application performance tools)
- Unit 11 - Security
- Lab 11 (Security)

Unit 1. Overview of DB2 9 on Linux, UNIX and Windows

What this unit is about

This unit describes the overall DB2 system, including a discussion on the components of the system. A high-level overview of the installation of DB2 is also covered.

The DB2 products that run on Windows and Linux/UNIX platforms are collectively known as *DB2 for Linux, UNIX, and Windows*. The rest of this document discusses DB2 on these platforms, unless otherwise noted. References to “DB2” refer to the DB2 Version 9.1. When a specific reference is needed to a distinct product, such as DB2 for AIX, DB2 for Windows, or DB2 for Linux/UNIX, it will be stated explicitly.

Also, there will be times when an indication will be made of a difference between Linux/UNIX and Windows. The comments indicated for Linux/UNIX apply to all Linux/UNIX products including those that run on Intel hardware. The comments indicated for Windows apply to Windows NT, Windows 2000, Windows XP, Windows ME, and Windows .NET.

What you should be able to do

After completing this unit, you should be able to:

- Contrast the DB2 Family of products
- Identify the DB2 Products
- Describe the functions of DB2 components
- Explore installation and parameters

How you will check your progress

Accountability:

- Machine exercises

References

Administration Guide

Quick Beginnings for DB2 Servers

Quick Beginnings for DB2 Clients

Getting started with DB2 installation and administration on Linux and Windows

<http://www.ibm.com/software/data/db2/udb/support/manualsv9.html>

Unit objectives

After completing this unit, you should be able to:

- Contrast the DB2 family of products
- Identify the DB2 products
- Describe the functions of DB2 components
- Explore installation and parameters

© Copyright IBM Corporation 2007

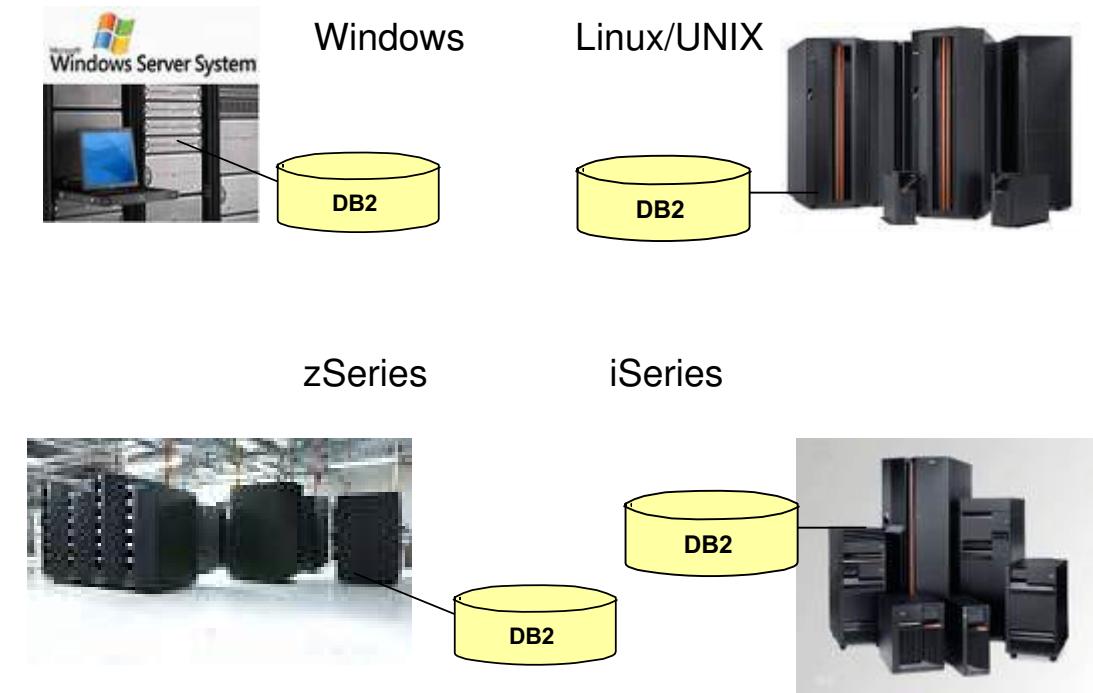
Figure 1-1. Unit objectives

CF238.3

Notes:

1.1 Overview of DB2

DB2 products



© Copyright IBM Corporation 2007

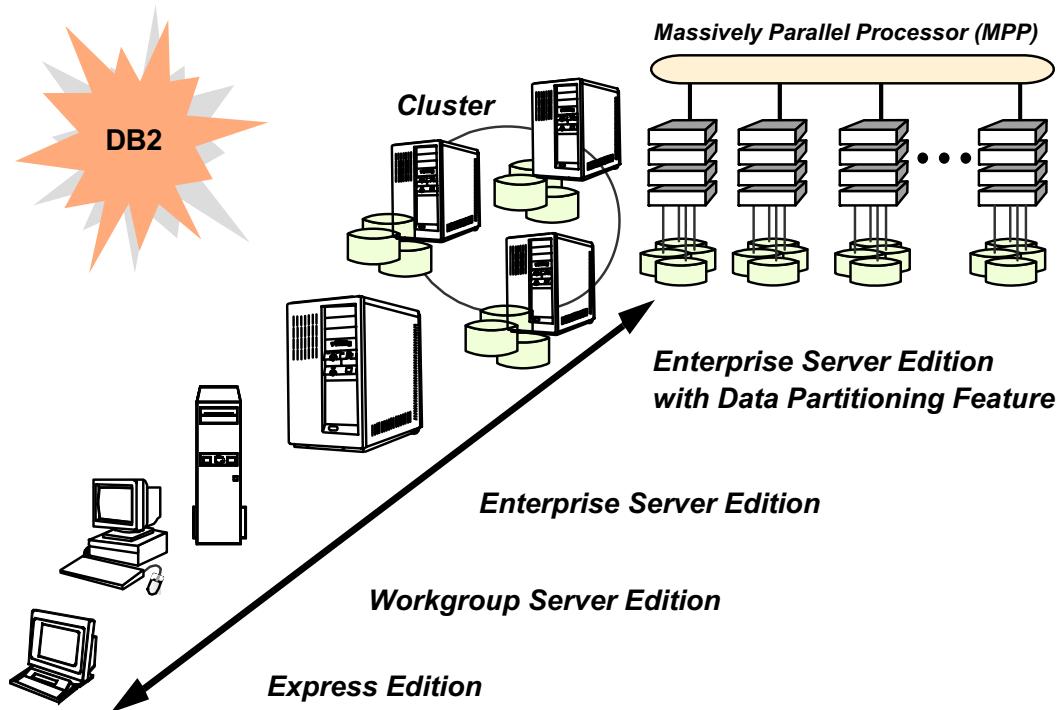
Figure 1-2. DB2 products

CF238.3

Notes:

DB2 is a relational database management system (RDBMS) that enables users to create, update, and control relational databases using Structured Query Language (SQL). Designed to meet the information needs of small and large businesses alike, it is available on a variety of platforms, including large systems such as z/OS, OS/390, VM, and VSE; mid-sized systems such as iSeries, pSeries, Linux, HP-UX, and Solaris; and single or LAN-based systems such as Windows 2003 and Windows XP. Clients running V8 or V9 can access DB2 9 servers. Clients running Version 7 are supported with some restrictions. All members of the DB2 family have similar external architecture and use many of the same key algorithms, thereby ensuring portability for customers across platforms.

DB2 — The scalable DB



© Copyright IBM Corporation 2007

Figure 1-3. DB2 — The scalable DB

CF238.3

Notes:

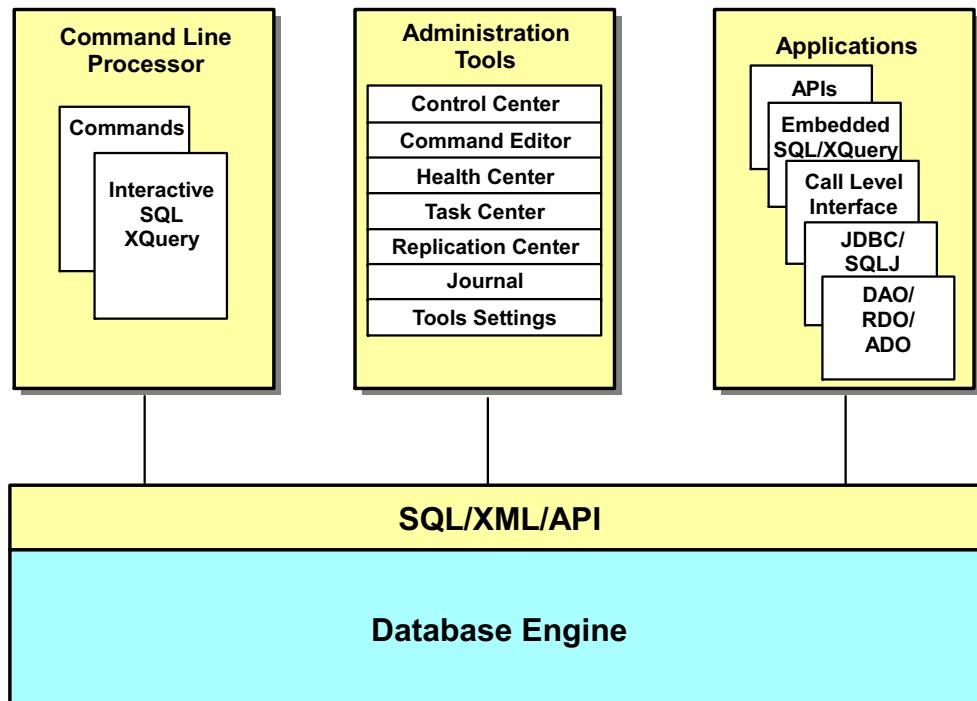
The graphic shows the scalability of the DB2 database server. DB2 is capable of supporting hardware platforms from uniprocessor laptops to massively parallel systems with hundreds of nodes and many processors per node. In between, it can support SMP machines or clusters of SMP machines. This provides both extensive and granular growth.

- DB2 Express Edition (EE) is the ideal entry level data server. Suitable for transaction processing or complex query workloads on servers with up to two processors. Your workload may be smaller, but your business data is as critical to you as to the largest enterprise. DB2 Express 9 provides many of Enterprise capabilities as value add features to allow you to control costs by purchasing only the capability you need. EE includes all core DB2 capabilities, packaged in two media packs — the Core CD and the ECF (electronic certification file), being for either Authorized User or CPU.
- DB2 Workgroup Server Edition (WSE) provides full DB2 database server capacity for your departmental LANs running Windows NT, Windows 2000, Windows XP, Windows .NET, AIX, HP-UX, Linux, and Solaris operating environments. It is the database server designed for deployment in a departmental or small business

environment that involves a small number of internal users. Workgroup Server Edition uses a licensing model designed to provide an attractive price point for smaller installations while still providing a full function database server. Workgroup Server Edition can be deployed in Linux, UNIX, and Windows server environments on systems with up to four CPUs. WSE is packaged in two media packs, which includes the Core CD and the ECF, being for either Authorized User or CPU.

- DB2 Enterprise Server Edition (ESE) meets the database server needs of midsize to large businesses. It can be deployed in Linux, UNIX, and Windows server environments on any size server. Enterprise Server Edition is the ideal foundation for building data warehouses, transaction processing, or Web-based solutions as well as a backend for packaged solutions like Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), and Supply Chain Management (SCM). Additionally, Enterprise Server Edition offers connectivity and integration for other enterprise DB2 and Informix data sources. Platforms supported include Windows NT, Windows 2000, Windows .NET, Linux, Linux/390, AIX, HP-UX, and Solaris. ESE is packaged in two media packs,, including the Core CD and the ECF (being for either Authorized User or CPU).
- DB2 Database Partitioning Feature (DPF) allows ESE customers to partition a database within a single system or across a cluster of system. The DPF capability provides the customer with multiple benefits, including scalability to support very large databases or complex workloads and increased parallelism for administration tasks. The DPF feature is only available in the CPU Option, contained in a separate ECF.
- The ECFs for each of the Features contain the entitlements for both the Authorized User and CPU Option, with the exception of DPF and Homogeneous Federation Feature, which are only available in the CPU Option.

Accessing DB2



© Copyright IBM Corporation 2007

Figure 1-4. Accessing DB2

CF238.3

Notes:

The main DB2 software components are:

- **Database Engine** — Provides the base functions of DB2. It manages and controls all access to data. It generates packages (stored access paths to the data), provides transaction management, ensures data integrity and data protection, and provides concurrency control.

The basic elements of the Database Engine are database objects, system catalogs, directories, and configuration files. All access to data takes place through the SQL or XML interface. For all commands the API interface is used.

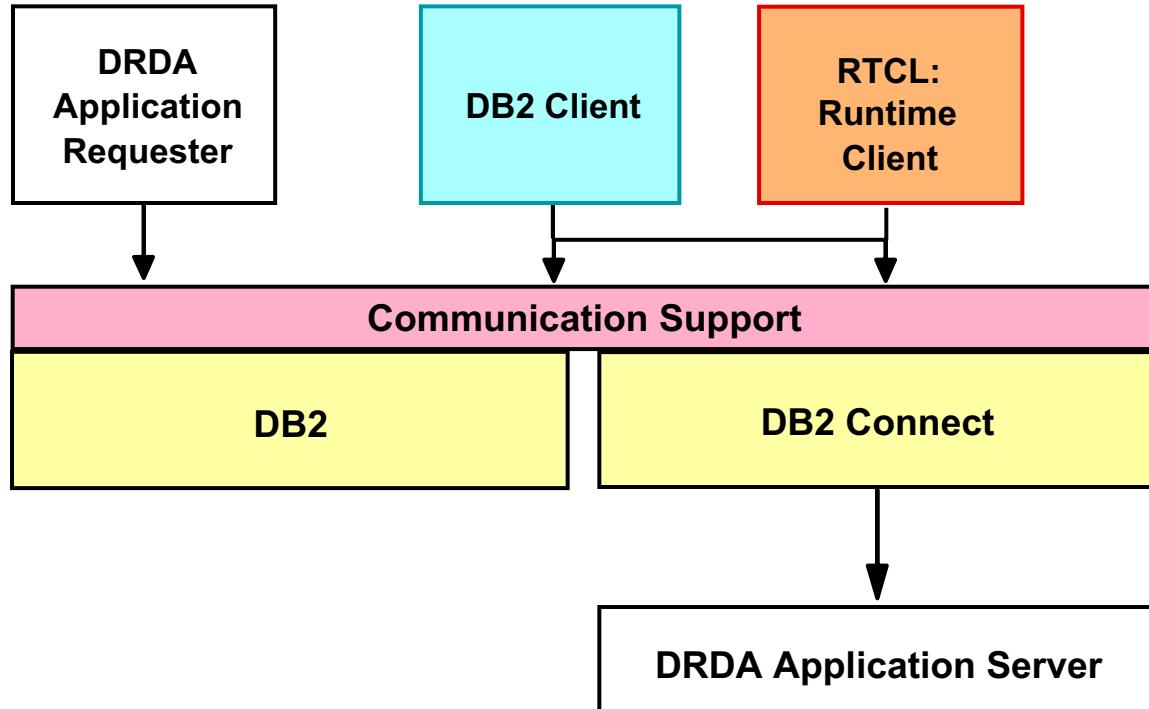
- **Command Line Processor (CLP)** — CLP is used to dynamically execute SQL requests, XQuery requests, DB2 commands, or all of them. CLP may be used to access local workstation databases, remote workstation databases, or remote Distributed Relational Database Architecture Application Server (DRDA AS) databases via DB2 Connect Personal Edition or DB2 Connect Enterprise Edition.

- **Administration Tools** — A collection of GUI tools that help to manage and administer databases, including:
 - **Control Center** — Manage systems, DB2 instances, DB2 for OS/390 and z/OS subsystems, databases, and database objects such as tables and views; display all of your systems, databases, and database objects and perform administration tasks on them; from the Control Center, open other centers and tools to help you optimize queries, jobs, and scripts, perform data warehousing tasks, create stored procedures, and work with DB2 commands.
 - **Replication Center** — Administer relational data between DB2 servers or databases; define replication environments, copy designated changes from one location to another, and synchronize the data in both locations.
 - **Command Editor** — Execute DB2 commands, XQuery and SQL statements, execute z/OS or OS/390 host system console commands, work with command scripts, and view a graphical representation of the access plan for explained SQL and XQuery statements.
 - **Task Center** — Organize task flow, schedule tasks, and distribute notifications about the status of completed tasks. A task can be created from a script that contains DB2, operating system, or MVS JCL commands.
 - **Health Center** — Identifies key performance and resource allocation problems within DB2, through notifications such as alarms or warnings, and provides recommended actions that can help to resolve the problems.
 - **Journal** — Focal point for viewing historical information generated within the Control Center and its components; views include: Task History, Database History, Performance Monitor, Alerts, Messages, Notification Log messages.
 - **License Center** — Display license status and usage information for DB2 products installed on your system; configure your system for license monitoring.
 - **Development Center** — Work with stored procedures, user-defined functions (UDFs), and structured types; modify existing routines and rebuild them, and run the routines on the database server for testing and debugging; deploy routines from a development project or database to a production server, and export or import routines to and from your project.
 - **Tools Settings** — For setting termination characters, fonts, OS/390 and z/OS options, Health Center Status Beacon options, and scheduler settings.
- **Applications** — To access the database, you can:
 - Use DB2 APIs to perform administrative functions such as backing up and restoring databases.
 - Embed static and dynamic SQL statements as well as XQuery statements in your applications.
 - Code DB2 Call Level Interface (DB2 CLI) function calls in your applications to invoke dynamic SQL statements. DB2 CLI is based on the Microsoft Open Database Connectivity (ODBC) specification, and the ISO CLI standard.
 - Develop Java applications and applets that call the Java Database Connectivity application programming interface (JDBC API). DB2 implements two

standards-based Java programming APIs: Java Database Connectivity (JDBC) and embedded SQL for Java (SQLJ).

- Develop Microsoft Visual Basic and Visual C++ applications that conform to Data Access Object (DAO) and Remote Data Object (RDO) specifications, and ActiveX Data Object (ADO) applications that use the Object Linking and Embedding Database (OLE DB) Bridge.
- Develop applications using IBM or third-party tools such as Net.Data, Excel, Perl, and Open Database Connectivity (ODBC) end-user tools such as Lotus Approach, and its programming language, LotusScript.

DB2 product components



© Copyright IBM Corporation 2007

Figure 1-5. DB2 product components

CF238.3

Notes:

DB2 provides a full-function, robust, relational database management system (RDBMS) along with a set of related products designed for the DB2 platforms. DB2 brings to these platforms the support of the IBM SQL technology that is available on all members of the DB2 family.

A Communication Support product must be installed on the server to enable remote client access.

Distributed Relational Database Architecture (DRDA) Application Requesters represent DRDA requests coming from, for example, z/OS, VM, and OS/400. No special product or enablement, other than support of remote clients, is required on the DB2 Server to support DRDA Application Requesters.

The DB2 Application Development Client and DB2 Administration Client have been combined and renamed to DB2 Client.

The DB2 Client provides the ability for workstations from a variety of platforms to access and administer DB2 databases. The DB2 Client has all the features of the DB2 Runtime

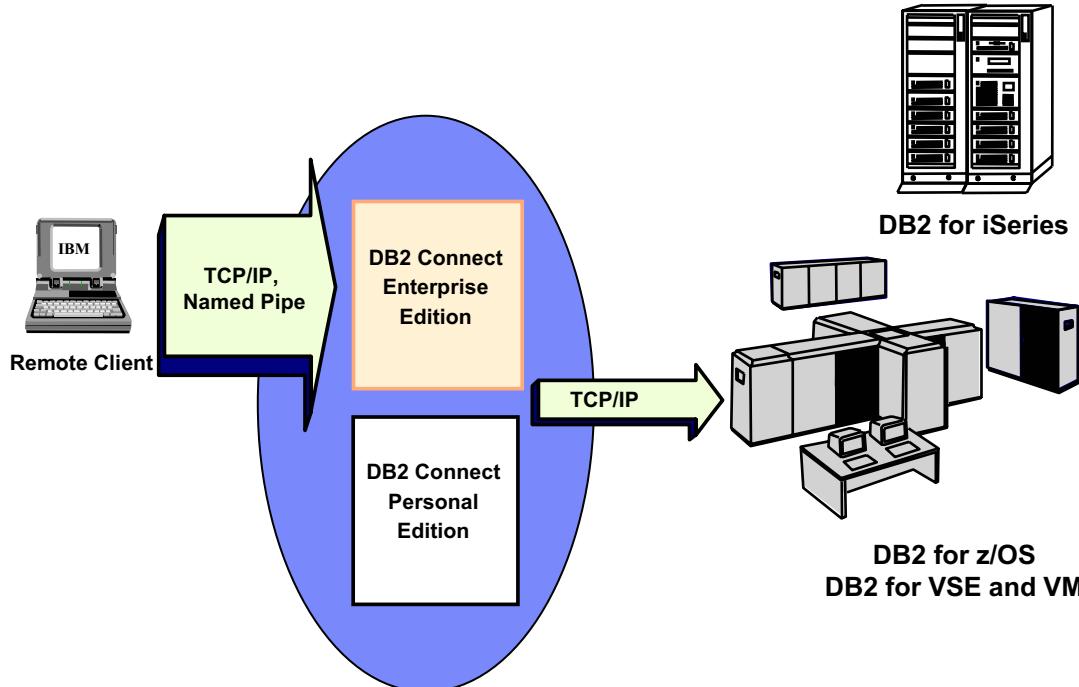
Client and also includes all the DB2 administration tools and support for Thin Clients. The DB2 Client is a collection of graphical and non-graphical tools and components for developing character-based, multimedia, and object-oriented applications. Special features include a plug-in to the Developer Workbench (was Development Center) and sample applications for all supported programming languages.

The DB2 Runtime Client (RTCL) is a lightweight client that provides the functionality required for an application to access DB2 servers and DB2 Connect servers. Functionality includes communication protocol support and support for application interfaces such as JDBC, SQLJ, ODBC, CLI, and OLE DB. Compared to previous versions, due to the removal of most of the previous Run-Time Client GUI facilities, the current Runtime Client has diminished disk requirements.

DB2 Connect provides DRDA application requester (AR) function. With Communications Support, DB2 Connect enables applications running on DB2 for Linux, UNIX, and Windows platforms to access and update data on DB2 for OS/390, DB2 for z/OS, DB2 for VM and VSE, DB2 for iSeries, and other database management systems that are DRDA-compliant. The DB2 Connect product incorporates the following utilities and functions of DB2:

- Command Line Processor (CLP), which allows you to issue SQL statements to access host platforms
- Database system monitor utility, which is an aid in the problem determination area
- The bind command, which allows you to bind packages to the host platforms
- The IMPORT and EXPORT utilities, which allow you to move data between host platforms and the DB2 Connect workstations
- The ability to catalog databases for the host platform

DRDA Application Requester (AR) support



© Copyright IBM Corporation 2007

Figure 1-6. DRDA Application Requester (AR) support

CF238.3

Notes:

DRDA Application Requester (AR) support is provided by DB2 Connect. DB2 Connect is available either in DB2 Connect Personal Edition, DB2 Connect Enterprise Edition, or DB2 Connect Unlimited Edition.

DB2 Connect Personal Edition provides access from a single workstation to DB2 databases residing on servers such as OS/390, z/OS, OS/400, VM, and VSE, as well as to DB2 servers on Linux/UNIX and Windows operating systems. DB2 Connect Personal Edition provides the same rich set of APIs as DB2 Connect Enterprise Edition. This product is currently available for Linux and Windows operating systems.

DB2 Connect Personal Edition is used to connect a single Windows operating system, or Linux workstation, to a host or iSeries database. DB2 Connect Personal Edition is best suited for environments where native TCP/IP support is provided by the database servers, and the application being deployed is a traditional 2-tier client/server application.

DB2 Connect Enterprise Edition is a connectivity server that concentrates and manages connections from multiple desktop clients and web applications to DB2 database servers running on host or iSeries systems. IBM's DB2 for iSeries, DB2 for OS/390 and z/OS, and

DB2 for VSE and VM databases continue to be the systems of choice for managing most critical data for the world's largest organizations. While these host and iSeries databases manage the data, there is a great demand to integrate this data with applications running on Windows and Linux/UNIX workstations.

DB2 Connect Enterprise Edition enables local and remote client applications to create, update, control, and manage DB2 databases and host systems using Structured Query Language (SQL), DB2 APIs (Application Programming Interfaces), ODBC (Open Database Connectivity), JDBC (Java Database Connectivity), SQLJ (Embedded SQLJ for Java), or DB2 CLI (Call Level Interface). In addition, DB2 Connect supports Microsoft Windows data interfaces such as ActiveX Data Objects (ADO), Remote Data Objects (RDO), and Object Linking and Embedding (OLE) DB.

DB2 Connect Enterprise Edition is currently available for AIX, HP-UX, Linux, Solaris, and Windows operating systems. These servers provide support for applications running on Linux, UNIX (AIX, HP-UX, and Solaris operating environments), and Windows workstations.

DB2 Connect Enterprise Edition is often installed on an intermediate server to connect DB2 clients to a host or iSeries database. It can also be used on machines where multiple local users want to access the host or iSeries servers directly.

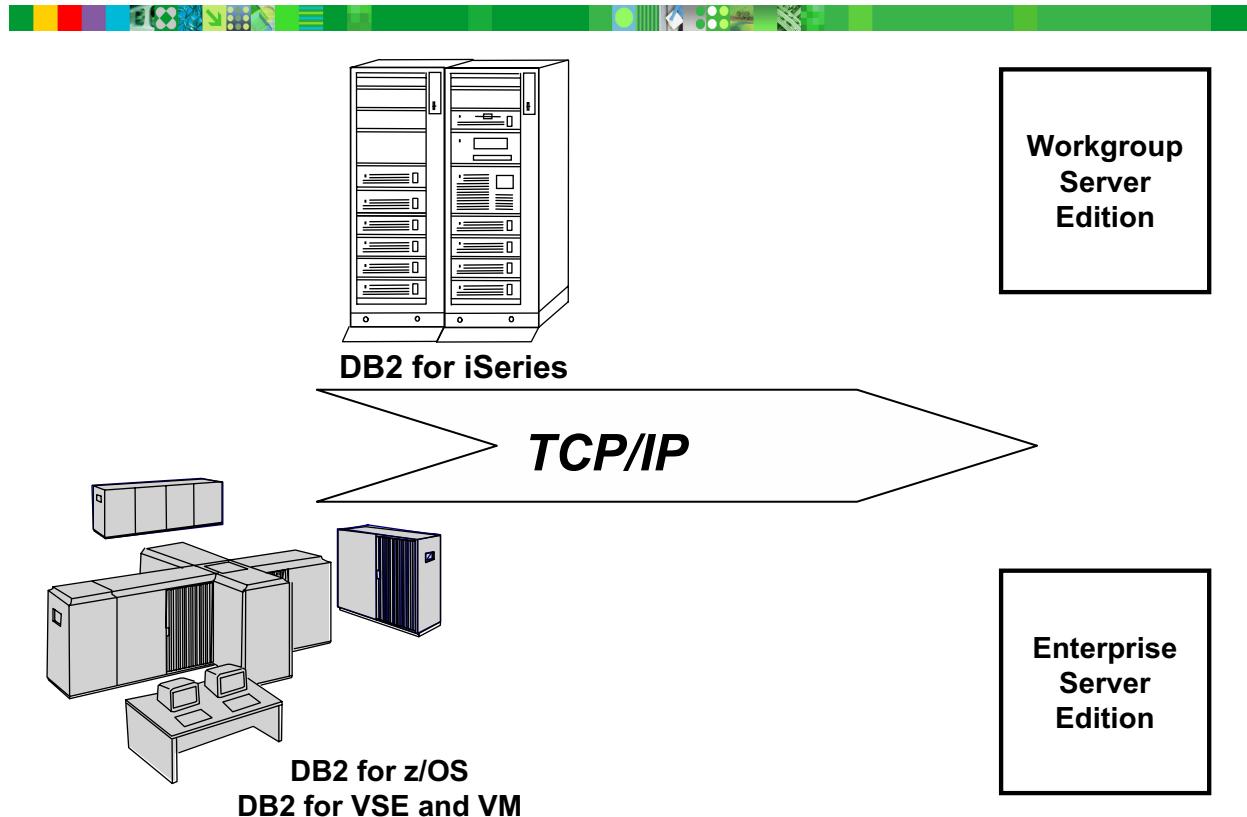
DB2 Connect Enterprise Edition is most appropriate for environments where:

- Web servers run Web-based applications.
- Web servers run Web-based application using data-aware Java applications.
- A middle-tier application server is used.
- TP monitors, such as CICS, Encina, Microsoft Transaction Server (MTS), Tuxedo, Component Broker, and MQSeries, are used.

DB2 Connect Unlimited Edition and DB2 Connect Application Server Edition are unique package offerings that allows complete flexibility of DB2 Connect deployment and simplifies product selection and licensing. This product contains both DB2 Connect Personal Edition and DB2 Connect Enterprise Edition with license terms and conditions that allow the unlimited deployment of any DB2 Connect product. License charges are based on the size of the S/390 or zSeries server that DB2 Connect users will be working with.

This package offering is only available for OS/390 and z/OS systems, and licensing is only valid for DB2 for OS/390 and z/OS data sources.

DRDA Application Server (AS) support



© Copyright IBM Corporation 2007

Figure 1-7. DRDA Application Server (AS) support

CF238.3

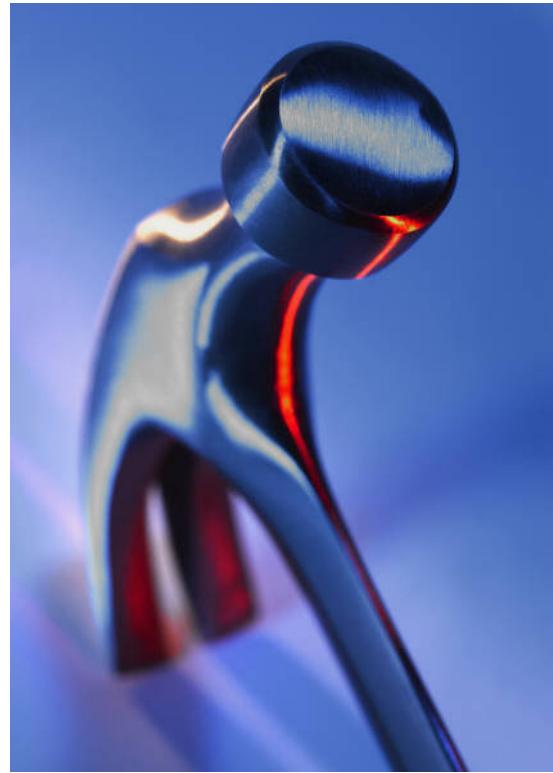
Notes:

DRDA Application Server (AS) support is provided in Workgroup Server Edition and Enterprise Server Edition. This allows a host client access to data on a DB2 server.

DRDA two-phase commit over TCP/IP is supported on all platforms for the DRDA AS.

Installation prerequisites

- Disk space
- Enough memory
- Available Internet browser
- Security (Authorities) concept for SYSADM / Admin-Server
- Specific requirements depending on operating system



© Copyright IBM Corporation 2007

Figure 1-8. Installation prerequisites

CF238.3

Notes:

The disk space required for your product depends on the type of installation you choose and the type of file system you have. The DB2 Setup wizard provides dynamic size estimates based on the components selected during a typical, compact, or custom installation.

At a minimum, a DB2 database system requires 256 MB of RAM. For a system running just DB2 and the DB2 GUI tools, a minimum of 512 MB of RAM is required. However, 1 GB of RAM is recommended for improved performance. These requirements do not include any additional memory requirements for other software that is running on your system.

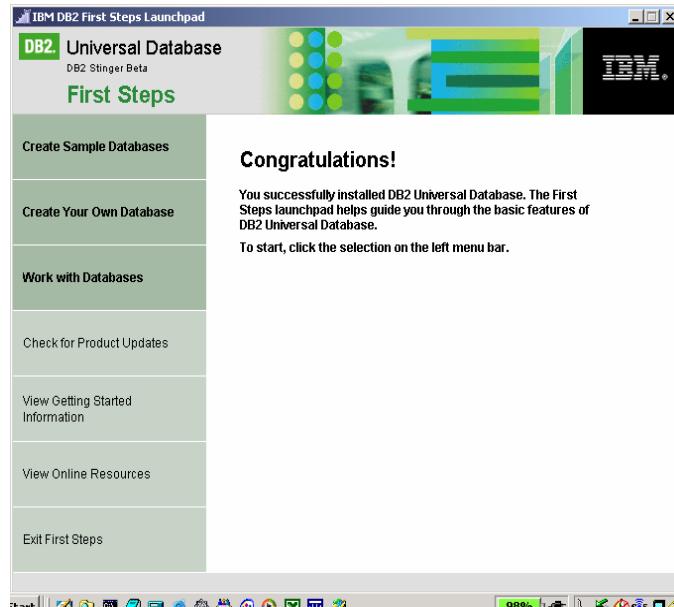
One of the following browsers is required to view online help, run the DB2 install launchpad (setup.exe), and to run First Steps (db2fs):— Internet Explorer 5.5 and up — Mozilla 1.4 and up — Firefox 1.0 and up — Netscape 7.0 and up.

Security considerations are important to the DB2 administrator from the moment the product is installed. To complete the installation of the DB2 database manager, a user ID, a group name, and a password are required. The GUI-based DB2 database manager install program creates default values for different user IDs and the group. Different defaults are created, depending on whether you are installing on Linux/UNIX or Windows platforms.

- For the most up-to-date prerequisite information, see
<http://www.ibm.com/software/data/db2/udb/sysreqs.html>.
- Getting Started Manuals

Installation steps

- GUI or non-GUI
- Customizing?
- Default instance?
- Admin-Server?
- Notification?
- Installation verification



© Copyright IBM Corporation 2007

Figure 1-9. Installation steps

CF238.3

Notes:

You can decide to run the installation with a command or using the GUI, where you will be guided step by step. You must have a local Administrator account (Windows) or root authority (Linux, UNIX) to successfully start the installation.

During both installation processes you can select typical or customized installation, whereby in the customized installation you would be able to select the components you want to install. Furthermore you need to decide during the installation if you want the installation process having create the DB2 default instance and the Administration server for you.

In an Windows environment the DB2 product will be installed, by default, in the `x:\Program Files\IBM\sql11b` directory, where x: represent the drive letter of the drive where you have installed your DB2 product.

If you are installing on a system where this directory is already being used, the DB2 product installation path will have `_xx` added to it, where `_xx` are digits, starting at 01 and increasing depending on how many DB2 copies you have installed. You can also specify your own DB2 product installation path.

For information on errors encountered during installation, review the installation log file located in the My Documents\DB2LOG\ directory. The log file uses the following format: **DB2-ProductAbbrev-DateTime.log**, for example, **DB2-ESE-Tue Apr 04 17_04_45 2006.log**.

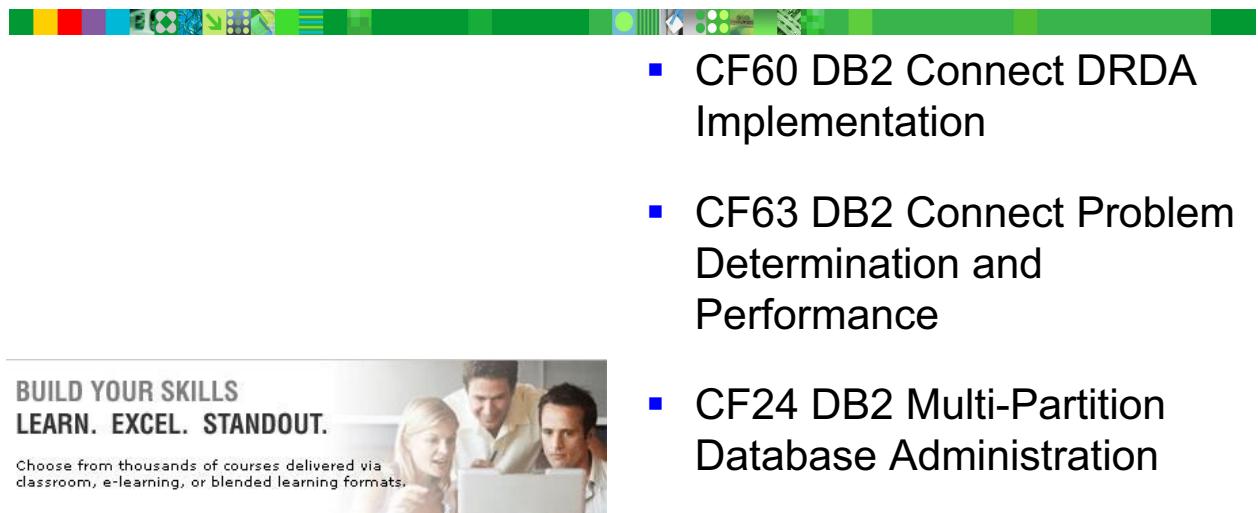
In an Linux environment the DB2 product will be installed, by default, in the following directory: **/opt/ibm/db2/V9.1**. Also here the same naming rules applies for installation of several DB2 copies, as well as that you can specify your own installation path.

The installation logs, **db2setup.log** and **db2setup.err** will be located, by default, in the **/tmp** directory. You can specify the location of the log files.

After verification of the relevant installation logs, you might need to update the relevant registries.

After a successful installation, you can use the FIRST STEPS GUI to create for example the SAMPLE database.

For more information



- CF60 DB2 Connect DRDA Implementation
- CF63 DB2 Connect Problem Determination and Performance
- CF24 DB2 Multi-Partition Database Administration
- CG24 DB2 Multi-Partitioned Environment for Single Partition DBAs

© Copyright IBM Corporation 2007

Figure 1-10. For more information

CF238.3

Notes:

You, or other people in your organization, may desire additional details. Please visit www.ibm.com/services/learning for the most current information and schedules of these courses in your country.

Unit summary

Having completed this unit, you should be able to:

- Contrast the DB2 family of products
- Identify the DB2 products
- Describe the functions of DB2 components
- Explore installation and parameters

© Copyright IBM Corporation 2007

Figure 1-11. Unit summary

CF238.3

Notes:

Unit 2. Command Line Processor (CLP) and GUI usage

What this unit is about

This unit explains the two methods of interacting with the DB2 database server system. Practice use of the CLP and GUI tools will be explored, using common database commands. The need for the DAS Administration Server will also be explained.

What you should be able to do

After completing this unit, you should be able to:

- Use the Command Line Processor
- Explore the GUI environment
- Describe the DAS role with GUI tools

How you will check your progress

Accountability:

- Machine exercises

Unit objectives



After completing this unit, you should be able to:

- Use the Command Line Processor
- Explore the GUI environment
- Describe the DAS role with GUI tools

© Copyright IBM Corporation 2007

Figure 2-1. Unit objectives

CF238.3

Notes:

2.1 Command Line Processor (CLP)

CLP Command Line Processor



```
DB2 CLP - DB2V9
Number of entries in the directory = 1
Database 1 entry:
Database alias          = MUSICDB
Database name           = MUSICDB
Local database directory = C:
Database release level = b.00
Comment                 =
Directory entry type   = Indirect
Catalog database partition number = 0
Alternate server hostname =
Alternate server port number =

C:\Program Files\IBM\SQLLIB_02\BIN>db2 connect to musicdb
      Database Connection Information
Database server        = DB2/NT 9.0.0
SQL authorization ID  = FC082139
Local database alias   = MUSICDB

C:\Program Files\IBM\SQLLIB_02\BIN>
```

© Copyright IBM Corporation 2007

Figure 2-2. CLP Command Line Processor

CF238.3

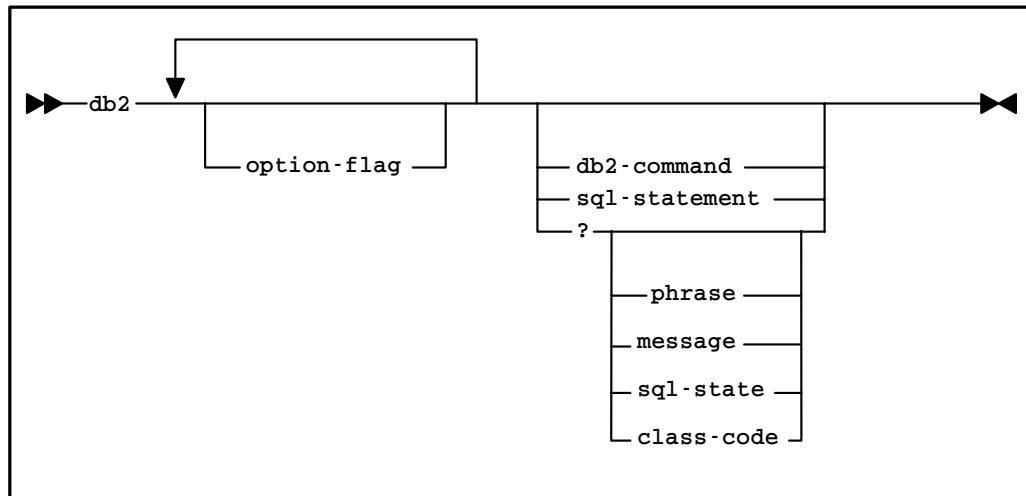
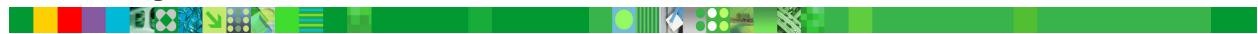
Notes:

Through the Command Line Processor you can issue:

- SQL statements
- XQuery statements (with the prefix XQUERY)
- DB2 commands
- OS system commands

In an Windows environment, the CLP can be found under the Command Line Processor as well as the *Command Window* and in an Linux/UNIX environment, it would be activated with the **db2profile**, so a *normal* terminal could be used.

CLP syntax



© Copyright IBM Corporation 2007

Figure 2-3. CLP syntax

CF238.3

Notes:

The options may be entered after the **db2** and before the command or statement. If the command or statement will not fit on one line, either:

- Continue typing and allow your typing to wrap and continue on the next line, or
- Use the \ (backslash) character for continuation.

The **class-code** option in the above syntax diagram means that you can request help for a message specified by a valid class-code.

Online reference



- Online Command Reference

```
db2 ?
db2 ? command string
db2 ? SQLnnnn      (nnnn = 4 or 5 digit SQLCODE)
db2 ? nnnnn       (nnnnn = 5 digit SQLSTATE)
```

- Online Reference Manuals

© Copyright IBM Corporation 2007

Figure 2-4. Online reference

CF238.3

Notes:

The Online Command Reference contains the syntax and explanations of all DB2 commands that may be executed through CLP. The Online Command Reference is invoked by:

1. **db2 ?** — List of all DB2 commands
2. **db2 ? command** — Information about specific commands
3. **db2 ? SQLnnnn** — Information about a specific SQLCODE generated by database manager. SQLCODE must be four or five digits in length.
4. **db2 ? nnnnn** — Information about a specific SQLSTATE generated by database manager. SQLSTATE must be five digits in length.

A set of reference manuals are also available online. For information on installation and setting up online reference manuals, consult your *Installing and Configuring DB2 Clients* manual.

Using the CLP



Non-interactive mode

```
db2 connect to musicdb
db2 "select * from syscat.tables" | more
    ↑(double quotes may be required)
```

Interactive mode

```
db2
db2=> connect to musicdb
db2=> select * from syscat.tables
```

© Copyright IBM Corporation 2007

Figure 2-5. Using the CLP

CF238.3

Notes:

Prefix all CLP commands or requests with **db2**, or use CLP in interactive mode by typing **db2** and then pressing Enter. In the interactive mode, the user can type CLP commands without prefixing them by db2.

When using CLP without being in interactive mode on a Linux/UNIX platform, remember to put quotes around the statement or command if special characters are included.

To issue XQuery statements in CLP, prefix the statements with the **XQUERY** keyword.

Interactive mode does not allow you to do piping or other operating system functions at the interactive CLP command level. To execute operating system commands without quitting interactive mode, issue **!<operating-system-command>**.

In the interactive mode, the history of commands is displayed by entering **history** (or shortened form **h**). Two variations are allowed:

- **reverse (r)** list in reverse order, in other words most recent first
- **num** (such as 5) limits the list to the last *num* history entries

Examples, assuming that you first entered db2start, then a “list active databases” command, and then you entered ? sql11111:

```
db2 => h
1 db2start
2 list active databases
3 ? sql11111
4 h
db2 => h r
5 h r
4 h
3 ? sql11111
2 list active databases
1 db2start
db2 => h r 3
6 h r 3
5 h r
4 h
```

There is also an edit (**e**) command which enables editing of a previous command. You can edit the command using the number in the history list, for example **e 2**. If no number is entered, the last command would be edited.

After having edited the command and closed the editor, the edited command is displayed in the CLP window, and you will be asked:

Do you want to execute the above command ? (y/n)

If you enter **y**, the command as edited is executed.

With **runcmd (r)** you can re-run a previously executed command. You can do this using the number from the history list, for example **r 3**. If no number is entered, the last command is re-executed.

If a command that you are entering exceeds the limit allowed at the command prompt, use a \ (backslash) as the line continuation character. If you want a blank space between the last character on the current line and the first character on the next line, don't forget to code a blank space before the backslash. You may wish to use the CLP flag **-t** instead of the \ (backslash). It says that the CLP statement terminates with a ; (semicolon).

In its output, CLP represents a SQL NULL value as a hyphen (-). If the column is numeric, the hyphen is placed at the right of the column. If the column is not numeric, the hyphen is at the left.

QUIT/TERMINATE/CONNECT RESET differences



<i>CLP COMMAND</i>	<i>Terminate CLP Back-end Process</i>	<i>Disconnect database Connection</i>
quit	No	No
terminate	Yes	Yes
connect reset	No	Yes if CONNECT=1 (RUOW)

© Copyright IBM Corporation 2007

Figure 2-6. QUIT/TERMINATE/CONNECT RESET differences

CF238.3

Notes:

To connect to a local or remote database, specify `db2 connect to dbname` where `dbname` maps to the alias name specified in the System Database Directory. After the `connect` is issued, all SQL requests are executed against the database to which you are connected.

If `CONNECT=1 (RUOW)`, **db2 connect reset** terminates the connection, and a subsequent SQL statement will cause connection to the default database, if it is defined.

The default database is defined in the environment variable DB2DBDFT. If `CONNECT=2 (DUOW)`, **db2 connect reset** puts the current connection in a dormant state and establishes a connection with the default database if it is defined. DUOW and CONNECT types will be defined in detail in the Distributed Management Topic.

db2 terminate issues a disconnect and also terminates the CLP back-end process.

The **quit** command ends the input mode and returns the user to the command prompt, but quit does not terminate CLP nor disconnect the database connection.

To end the CLP background process and disconnect the database connection issue the `terminate` command.

CLP command options



```
DB2 CLP - DB2V9

Option Description Current Setting
-a Display SQLCA OFF
-c Auto-Commit ON
-d Retrieve and display XML declarations OFF
-e Display SQLCODE/SQLSTATE OFF
-f Read from input file OFF
-i Display XML data with indentation OFF
-l Log commands in history file OFF
-m Display the number of rows affected OFF
-n Remove new line character OFF
-o Display output ON
-p Display interactive input prompt ON
-q Preserve whitespaces & linefeeds OFF
-r Save output to report file OFF
-s Stop execution on command error OFF
-t Set statement termination character OFF
-v Echo current command OFF
-w Display FETCH/SELECT warning messages ON
-x Suppress printing of column headings OFF
-z Save all output to output file OFF

C:\Program Files\IBM\SQLLIB_02\BIN>
```

db2 list command options

© Copyright IBM Corporation 2007

Figure 2-7. CLP command options

CF238.3

Notes:

Issue the **db2 list command options** command to view the current settings for the command line flags and the value of DB2OPTIONS.

The following shows the option flag, description, and default setting.

- **a** — Displays SQLCA data — OFF
- **c** — Autocommit SQL statements — ON
- **e{c | s}** — Display SQLCODE or SQLSTATE — OFF
- **f *filename*** — Read command input from a file instead of standard input — OFF
You may wish to issue *db2 < filename* instead of *db2 -f filename*.
- **i** — Display XML data with indentation — OFF
- **l *filename*** — Log commands in a history file — OFF
- **m** — Display the number of rows affected — OFF
- **n** — Remove new line character — OFF

- **o** — Display output data and messages to standard output — ON
- **q** — Preserver wildespaces and linfeeds — OFF
- **p** — Display CLP prompt when in interactive mode — ON
- **r *filename*** — Write output to a file — OFF
- **s** — Stop execution if errors occur while executing commands in a batch file or in interactive mode — OFF
- **t** — Set the statement termination symbol — OFF
- **v** — Echo command text to standard output — OFF
- **w** — Display FETCH/SELECT warning messages — ON
- **z *filename*** — Save all output to output file — OFF

CLP consists of two processes, the front-end process and the back-end process, which are required to maintain the database connection between each CLP invocation. These two processes communicate via three message queues: a request queue, an input queue and an output queue.

DB2BQTIME, **DB2BQTRY**, **DB2RQTIME**, and **DB2IQTIME** offer a means of configuring communication between the two processes. When CLP is invoked, the front-end process checks to see if the back-end process is already active. If it is, the front-end process reestablishes a connection with it. If not, the front-end process sleeps for the duration of time specified by the **DB2BQTIME** variable and then checks again. The front-end will perform this checking for the number of times specified by the **DB2BQTRY** variable. If the back-end process is still not active, it is activated.

Once the back-end process has been started, it waits on its request queue for a request from the front-end. It also waits on the request queue in between requests initiated from the command prompt. The **DB2RQTIME** variable specifies the length of time the back-end process waits for a request from the front-end process.

When the back-end process receives a request from the front-end process, it sends an acknowledgement to the front-end process indicating that it is ready to receive input via the input queue. The back-end process then waits on its input queue. The back-end process also waits on the input queue while a batch file is executing and while the user is in interactive mode. The **DB2IQTIME** variable specifies the length of time the back-end process waits on the input queue for the front-end process to pass commands.

Users can terminate the back-end process explicitly by issuing the TERMINATE command.

Modify CLP options



1 Temporary for Command

```
db2 -r options.rep list command options
db2 -svtf create.tab3
db2 +c "update tab3 set salary=salary + 100"
```

2 Temporary for Interactive CLP Session

db2=>update command options using c off a on

3 Temporary for non-interactive CLP Session

```
export DB2OPTIONS="-svt" (UNIX)
set DB2OPTIONS="-svt" (Windows)
db2 -f create.tab3
```

4 Every session

put point 3 in UNIX **db2profile**
or System Program Group in Windows

© Copyright IBM Corporation 2007

Figure 2-8. Modify CLP options

CF238.3

Notes:

The CLP options can be used in any sequence and combination. To turn the option on, prefix the option with a minus sign (-). To turn an option off, prefix the option with a minus sign (-) and follow the option letter with another minus sign (-) or prefix the option with a plus sign (+). For example, either use -c- or +c.

These options can also be specified by setting the DB2OPTIONS environment variable. ((Windows) set DB2OPTIONS='+c -a', (Linux/UNIX) export DB2OPTIONS='+c -a').

CLP option flags temporarily override DB2OPTIONS settings.

db2 update command options command allows the user to change an option setting from the interactive input mode or a command file.

db2 update command options using c off

Another way to redirect the CLP output from the screen to a file instead of using the **-r option** is to use the > symbol.

Some operating systems use the > or < symbols in SQL statements, and do not want them interpreted as redirection symbols by CLP. The quotation marks should surround all text to be processed by the CLP, but does not include any options.

- **db2 -r sample.rep “select * from org”**
- **db2 “select * from org where deptnumb > 38” > sample.rep**

This can be avoided by using a DB2 interactive CLP session.

Input file — no operating system commands



Edit create.tab

```
-- comment: db2 -svtf create.tab

connect to sample;

create table tab3
  (name varchar(20) not null,
   phone char(40),
   salary dec(7,2));

select * from tab3;

commit work;

connect reset;
```

db2 -svtf create.tab

© Copyright IBM Corporation 2007

Figure 2-9. Input file — no operating system commands

CF238.3

Notes:

Use an editor to create a file called **create.tab**.

Comments are denoted with a line that starts with two hyphens (--) .

A semicolon may be used to denote the end of a SQL statement if the file is executed with the **-t** command option.

In non-interactive mode, execute the file with **db2 -svtf create.tab**. Since *db2* was not coded in the input file, only DB2 commands may be coded in the input file. In non-interactive mode, the command to execute the commands in the input file must be started with *db2*. The **-s** option says to stop execution if an error occurs. The **-v** option says to echo the current command on the monitor screen. The **-t** option says the statements end with a *semicolon*. The **-f** option says the command input is read from an input file called **create.tab**.

Input file — operating system commands



vi seltab

```
echo "Table Name Is" $1 > out.sel
db2 "select * from $1" >> out.sel
```

edit seltab.cmd

```
echo 'Table Name Is' %1 > out.sel
db2 Select * from %1 >> out.sel
```

seltab.org

out.sel
contents

Table Name Is org

DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
10	Head Office	160	Corporate	New York
15	New England	50	Eastern	Boston
20	Mid Atlantic	10	Eastern	Washington
38	South Atlantic	30	Eastern	Atlanta
42	Great Lakes	100	Midwest	Chicago
51	Plains	140	Midwest	Dallas
66	Pacific	270	Western	San Francisco
84	Mountain	290	Western	Denver

© Copyright IBM Corporation 2007

Figure 2-10. Input file — operating system commands

CF238.3

Notes:

When including DB2 commands or SQL statements in an input file with operating system commands, place **db2** before the command or SQL statement and enclose the command or statement in double quotation marks.

db2 -r out.sel “select * from \$1” may also be used to redirect the output to the file **out.sel**. The output will still echo on the screen unless the **+o** option is used. The **-r** option does not append output to the file.

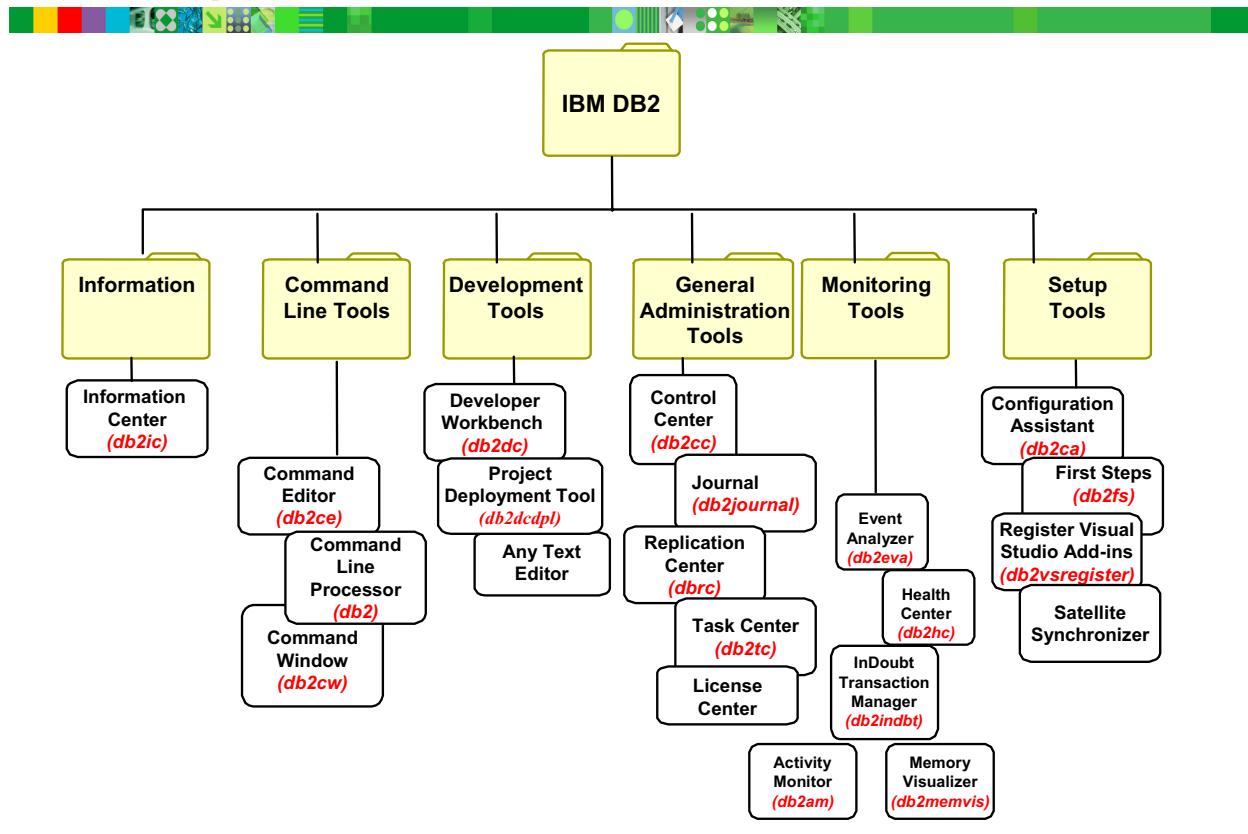
In Linux/UNIX, the user must have execute authority to execute the CLP input file. The following steps may be executed to do so:

```
$chmod 744 sel.tab
$ls -l sel.tab
```

```
-rwxr--r-- 1 inst31 adm31 58 Jul 15 13:53 sel.tab
```

2.2 Getting started with DB2 GUI tools

Roadmap to the GUI tools



© Copyright IBM Corporation 2007

Figure 2-11. Roadmap to the GUI tools

CF238.3

Notes:

The graphic shows an overall view, or roadmap, for accessing the graphical tools.

There are some tools that can be started from a command prompt, in addition to starting them from the GUI interface. These are shown on the graphic, if they exist, in italics and red color. For example, by entering the command `db2cmd` from a command prompt, you can start the DB2 Command Line Processor. The Configuration Assistant (CA) can be started using the command `db2ca`, as well as from the DB2 Desktop folder.

You can access many of the other graphical tools from within the Control Center. Most of the graphical tools have a menu bar option, Tools, and an icon bar, that will allow you to easily access any of the other graphical tools.

Control Center

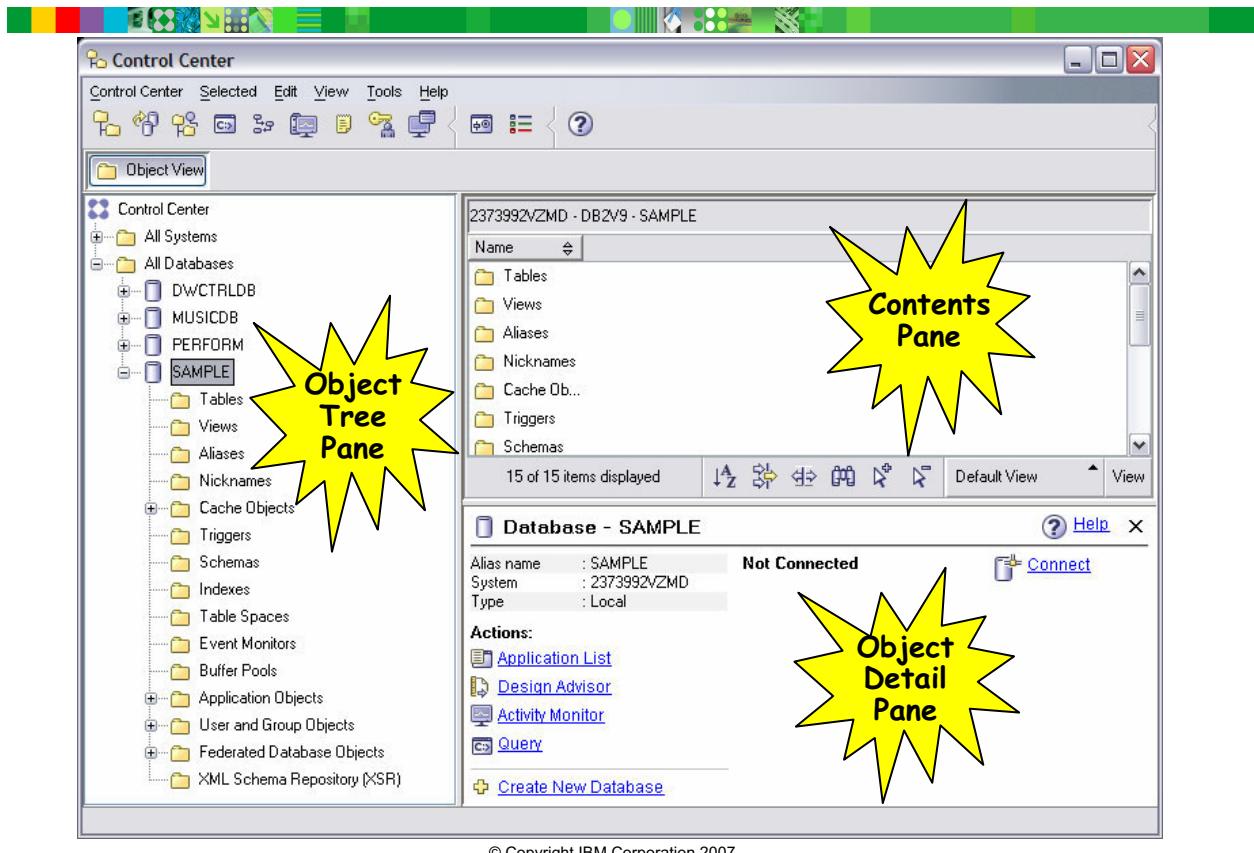


Figure 2-12. Control Center

CF238.3

Notes:

The Control Center is the central point of administration for DB2. There are three views that you can chose from:

- **Basic View:** Is a simplified view of DB2 core functionality, and omits features that are generally used by the DB2 system administrator but not by the DBA. This includes essential objects such as databases, tables, views, triggers, and stored procedures.
- **Advanced View:** Is the default and lists the standard set of objects.
- **Custom View:** Here you can select those features which you wish to see.

The Control Center is divided into three areas which have contents depending on your chosen view:

- **Object tree Pane**
- **Contents Pane**, showing the contents of the selected object.
- **Object Detail Pane**, which provides context-sensitive access to the main features of the object focused and selected in the other panes, like, for example, the Backup Database tool, the Storage Management tool, or Automatic Maintenance.

Use the Control Center to manage systems (different OSs), DB2 instances, DB2 for OS/390 and z/OS subsystems, databases, and database objects such as tables and views.

In general: The most often used parameters and options are included in UIF panels. If a special parameter or option needs to be used, use the command line interface instead.

In the Control Center, you can administer all of your systems, databases, and database objects. From the Control Center, you can also open other centers and tools to help you optimize queries, jobs, and scripts; perform data warehousing tasks; create stored procedures or UDFs; and work with DB2 commands.

The following are some of the key tasks that you can perform with the Control Center:

- Add DB2 systems, federated systems, DB2 for z/OS and OS/390 systems, instances, databases, and database objects to the object tree.
- Manage database objects. You can create, alter, and drop databases, table spaces, tables, views, indexes, triggers, and schemas. You can also manage users.
- Manage data. You can load, import, export, and reorganize data. You can also gather statistics.
- Perform preventive maintenance by backing up and restoring databases or table spaces.
- Monitor performance and troubleshoot problems.
- Configure and tune instances and databases.
- Manage database connections, such as DB2 Connect servers and subsystems.
- Manage applications.
- Analyze queries using Visual Explain to look at access plans.
- Launch other tools such as the Command Editor and the Health Center.
- Automatic maintenance of your databases.

In many cases, wizards and launchpads are available to help you perform these tasks more quickly and easily. In many cases, the wizard will launch automatically when you select a task to be performed on an object; for example, the Create action on the Tables object will launch the Create Table wizard.

Other components of the Control Center are the following:

- Menu Bar — It is used to access Control Center functions and online help using menus.
- Tool Bar — These icons are used to access all the other administration tools.
- Hover Help — Provides a short description for each icon on the toolbars.

The Systems object represents machines. You can see all of the systems that your system has catalogued by looking at the list in the Control Center. The local workstation is represented by a System icon labeled with the machine name of the local workstation; other remote systems defined are labeled as they are cataloged.

To display the instances on each system, expand the object tree by clicking the plus sign (+) on the local or remote system name.

Command Editor

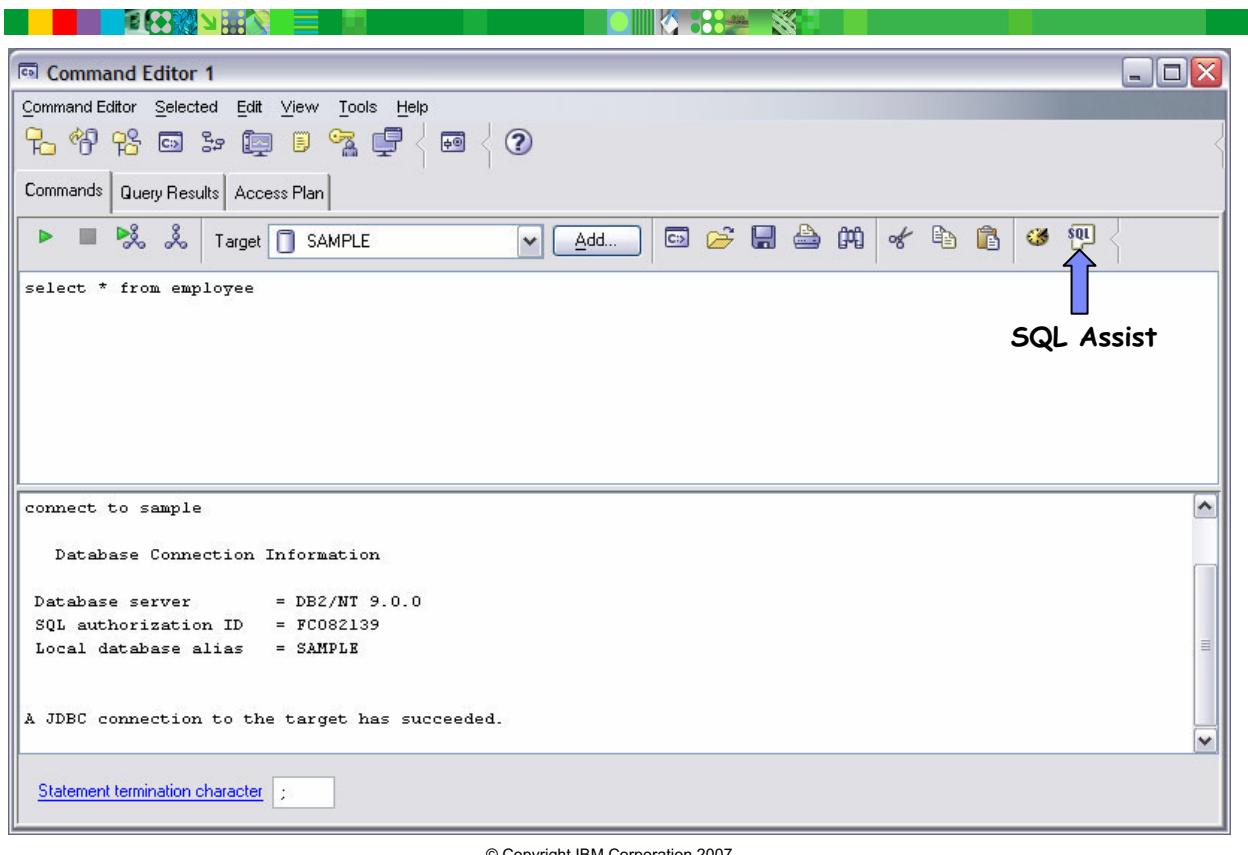


Figure 2-13. Command Editor

CF238.3

Notes:

Use the Command Editor to execute DB2 commands and SQL/XQUERY statements; to execute MVS, OS/390, or z/OS commands; to work with command scripts; and to view a graphical representation of the access plan for explained SQL statements.

On the **command** page, you can perform the following actions:

- Execute an SQL statement or DB2 CLP command. You do not need to precede the command by *DB2*.
- Run an existing script, by clicking the Execute icon (at the left of the tool bar).
- Execute the commands in sequence.
- Create and save a script. You can optionally store a saved script in the Task Center, where you can schedule the script to run at a specific time.
- Run and/or modify an existing script.

On the **Query Results** page, you can see the results of the queries. You can also save the results or edit the contents of the table.

On the **Access Plan** page, you can see the access plan for any explainable statement that you specified on the Interactive page or the Script page. DB2 generates the access plan when it compiles the SQL or XQUERY statement. You can use this information to tune your queries for better performance.

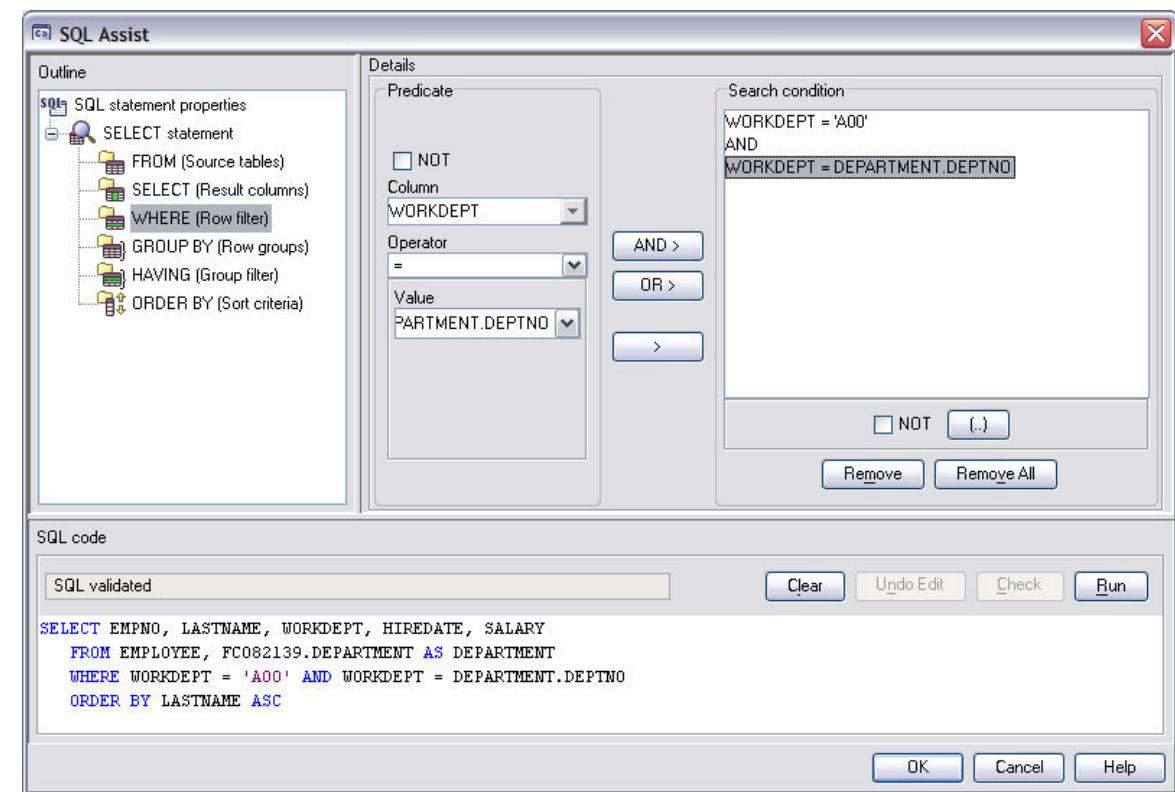
To invoke the SQL Assist tool, click the **SQL Assist** button on the Interactive page. To invoke the Visual Explain tool, execute an explainable statement on the Interactive page or the Script page.

The Web version of the Command Editor includes most of these features but it *does not* currently include Visual Explain or SQL Assist.

Both the GUI and traditional CLP are installed and shown as icons in the DB2 folder. For Windows, there is also a Command Window to enable users to execute SQL statements and DB2 commands from an MS-DOS command prompt.

The example commands shown in the Command Editor assume that a semi-colon (;) has been set in the Tools (can be specified at the bottom of the command editor window) as the statement termination character.

SQL Assist window



© Copyright IBM Corporation 2007

Figure 2-14. SQL Assist window

CF238.3

Notes:

Use SQL Assist to create SQL statements. With SQL Assist and some knowledge of SQL, you can create SQL SELECT statements. In some environments, you can also use SQL Assist to create INSERT, UPDATE, or DELETE statements.

The SQL Assist main window comprises three areas:

- Outline view
- Details view
- SQL code view

The Outline view contains a high-level representation of the current SQL statement. With the Outline view, you can visually examine an outline of the SQL statement and navigate through the steps of building the SQL statement. Select a node in the Outline view to view details for this element in the Details area.

Use the Details area to add elements to the SQL statement. The Details area changes based on what node you select in the Outline view. When you select a node in the Outline

view and make changes in the Details area, the SQL code is generated in the SQL code view.

The SQL code view contains the SQL code that is generated based on the contents of the Outline view and changes that you made in the Details area. The code is syntax-highlighted. In some environments, you can edit the SQL code in the SQL code view.

Refer to the Online Help for more information.

Administration tools

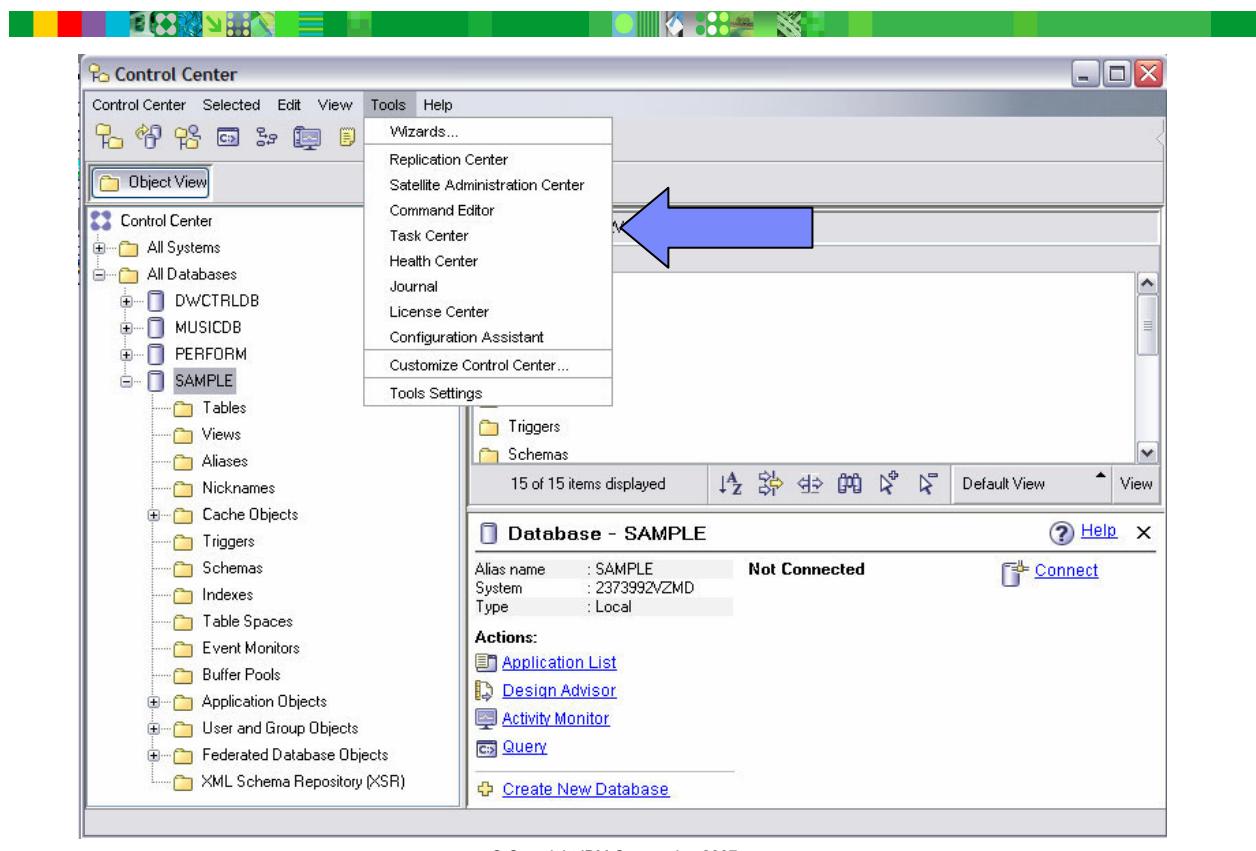


Figure 2-15. Administration tools

CF238.3

Notes:

From the DB2 Desktop folder, you can access the Control Center. From the Control Center, there is a set of database server administration tools, designed to help a database administrator manage and administer DB2 databases.

The administration tools consist of the following:

- Wizards that helps you perform tasks like backups, creation of tables or tablespaces, and so on.
- Control Center — The primary point of control for the administrator.
- Replication Center — To administer replication between a DB2 database and another relational database.
- Satellite Administration Center — To set up and administer a group of DB2 servers that perform the same business function.
- Command Editor — The same GUI command and statement processor available from the DB2 Desktop Folder.
- Task Center — To run tasks, either immediately or according to a schedule, and to notify people about the status of completed tasks.

- Health Center — To monitor the state of the database environment and make any necessary changes.
- Journal — Displays historical information about tasks, database actions and operations, Control Center actions, messages, and alerts.
- License Center — To display license status and usage information for DB2 products that are installed on the system.
- Development Center — To develop stored procedures, user-defined functions (UDFs), and structured types.
- Tools Settings — To modify the interface with the GUI tools.
- Configuration Assistant
- Customize Control Center

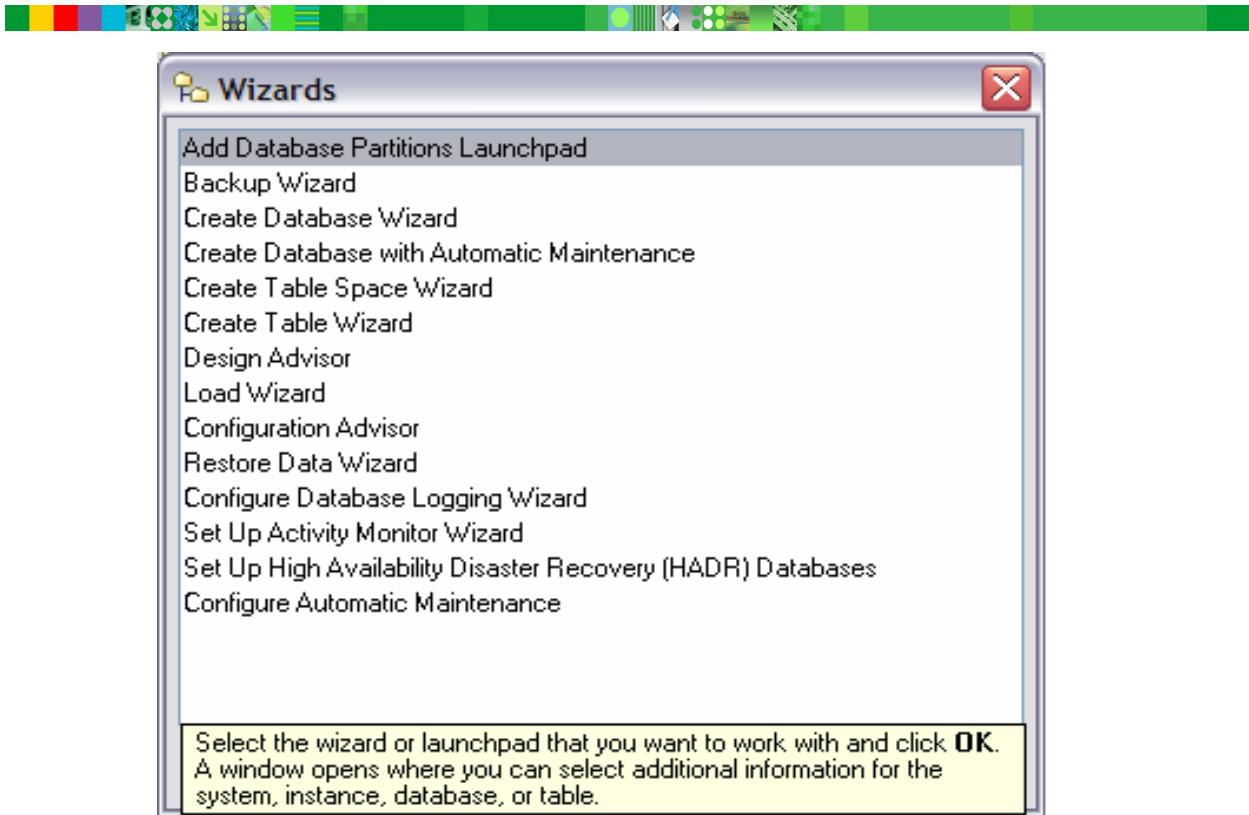
The administration tools are part of the Administration Client installation.

If installed on a remote client, the administration tools can be used from the remote machine to configure a dedicated database administrator's system. This allows remote administration of DB2 databases on all DB2 server platforms.

Some of these administration tools are explained in detail in the next several pages.

All dialogs that run from the Control Center that generate an SQL statement or a command have a button (option) to SHOW SQL or SHOW COMMAND. A displayed statement or command may be saved as a script.

Using the wizards



© Copyright IBM Corporation 2007

Figure 2-16. Using the wizards

CF238.3

Notes:

Wizards, Advisors, and Launchpads are integrated into the administration tools.

Wizards assist you in completing a single task by stepping you through the task. The wizard task overview on the first page of the wizard lists any prerequisite steps and briefly describes every page of the wizard. Other pages of the wizard may contain links to conceptual or reference information to help you understand the function of the wizard.

Advisors assist you with more complex tasks, such as tuning and performance tasks, by gathering information and recommending options that you might not consider. You can accept or reject the advice of the advisor. Advisors can be called from the GUI as well as from APIs and the command line interface. Conceptual and reference information is available to help you understand the function of the advisor.

Launchpads assist you in completing high-level tasks by stepping you through a set of tasks in the right order. Launchpads can call wizards or other dialogs to accomplish the high-level task. The launchpad task overview on the first page of the launchpad lists any prerequisite steps and briefly describes every page of the launchpad. Other pages of the

launchpad may contain links to conceptual or reference information to help you understand the function of the launchpad.

To select a wizard, advisor, or launchpad, from the Control Center window, select **Tools —> Wizards**. The **Wizards** window opens. Select the wizard, advisor, or launchpad you want to use. Select the object for which you want help and follow the instructions to complete the task.

The following wizards are available:

Create Database Wizard

Use the Create Database wizard to create a database with default or tailored settings.

- To create a database with default settings, complete only the first page of the wizard.
- To create a database with specific table spaces (containers), custom performance parameters, and local language attributes, complete all the pages of the wizard.
- To create a database with automatic maintenance, use the Create Database with automatic Maintenance wizard.

Create Table Space Wizard

Use the Create Table Space wizard to create new table spaces. First you name your table space and specify the type of table space you want to create, then you make more advanced choices.

Create Table Wizard

Use the Create Table wizard to create new tables in a database. First you name the table, determine the columns that you want, and specify space for storing the table data. From there, you continue on to make selections about keys, dimensions, and constraints.

Backup Database Wizard

Use the Backup wizard to back up the objects in a database or database partition. First you specify if you want to back up the entire database or database partition, or if you want to back up only selected table spaces or table space partitions, then you make more advanced choices.

Restore Database Wizard

Use the Restore wizard to perform any of the following tasks:

- Restore a database or database partition
- Roll forward a database or database partition
- Restore a database backup image to a new database
- Restore the history file for a database
- Restore a table space or table space partition
- Roll forward a table space or table space partition

First you specify the objects you want to restore and select the image that you want to use, then you make more advanced choices.

Load Wizard

Use the Load wizard to load data into a selected table. The Load wizard guides you through load configuration and the selection of options. The Load wizard also lets you copy an existing load task and use the values that are set in the existing load task for your new load task.

If you want to use exception tables with the load, you must create the exception tables before running the load task.

Design Advisor

Use the Design Advisor to optimize workload performance by suggesting a set of indexes. This advisor recommends which DB2 objects to create in your database to improve performance for a given set of SQL statements in a workload.

Configuration Advisor

Use the Configure Performance wizard to calculate suggested configuration parameters. A series of pages collects information about the server, workload, transaction type, priority, population characteristic, number of connections, and isolation level.

Configure Database Logging

Use the Configure Database Logging wizard to configure data logging options for a database. First, you specify whether you want to use circular logging or archive logging; then, you indicate how you want to handle your log files; then, you make more advanced choices.

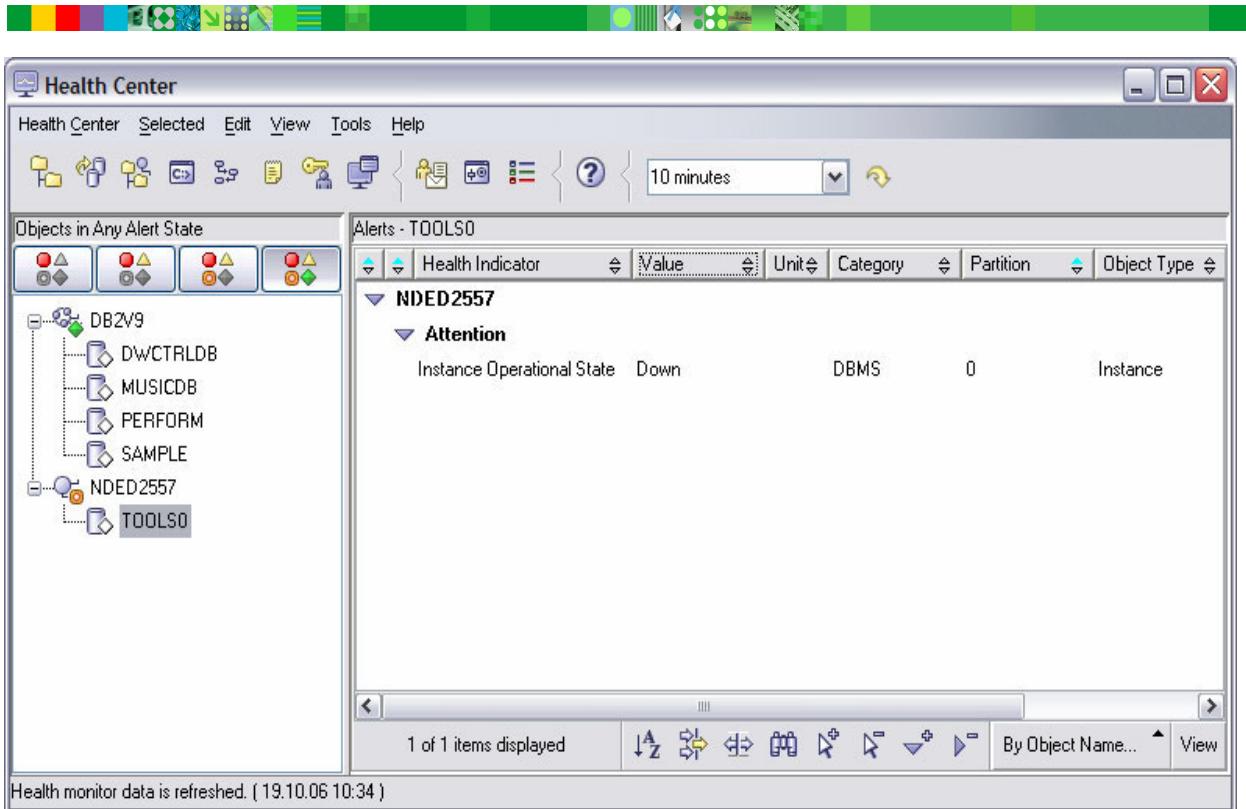
Add Partitions Launchpad

This launchpad provides access to the Add Partitions wizard and the Redistribute Data wizard.

Use the Add Partitions wizard to create a partition and add it to one or more database partition groups. First you add a new partition to your instance and assign the partition to one or more database partition groups, then you make more advanced choices.

Use the Redistribute Data wizard to create an effective redistribution plan for your database partition group and redistribute your data. First, you select your redistribution method and strategy, then you make more advanced choices.

Health Center



© Copyright IBM Corporation 2007

Figure 2-17. Health Center

CF238.3

Notes:

Use the Health Center to monitor the state of the database environment and make any necessary changes.

When you use DB2, a health monitor continuously monitors a set of health indicators. If the current value of a health indicator is worse than the appropriate threshold value, the health monitor generates an alert. DB2 comes with a set of predefined threshold values, which you can later customize. For example, you can customize the alarm and warning thresholds for the amount of free memory on the system.

Depending on your configuration, the following actions can occur when the health monitor generates an alert:

- An entry is written in the administration notification log, which you can read from the Journal.
- The health center status beacon appears in the lower right corner of the DB2 GUI Tools window.
- An e-mail or pager message is sent to the contacts that you specify for this instance.

The following are some of the key tasks that you can perform with the Health Center:

- View the status of the database environment. Beside each object in the navigation tree, an icon indicates the most severe alert for the object (or for any objects contained by that object). For example, a green diamond icon beside an instance means that the instance and the databases contained in the instance do not have any alerts.
- View the alerts for an instance or a database. When you select an object in the navigation tree, the alerts for that object are shown in the pane to the right.
- View detailed information about an alert, and recommended actions. When you double-click an alert, a notebook appears. The first page shows the details for the alert. The second page shows any recommended actions.
- Configure the health monitor settings for a specific object, and the default settings for an object type or for all objects within an instance.
- Select which contacts will be notified of alerts.
- Review the history of alerts for an instance.

Journal

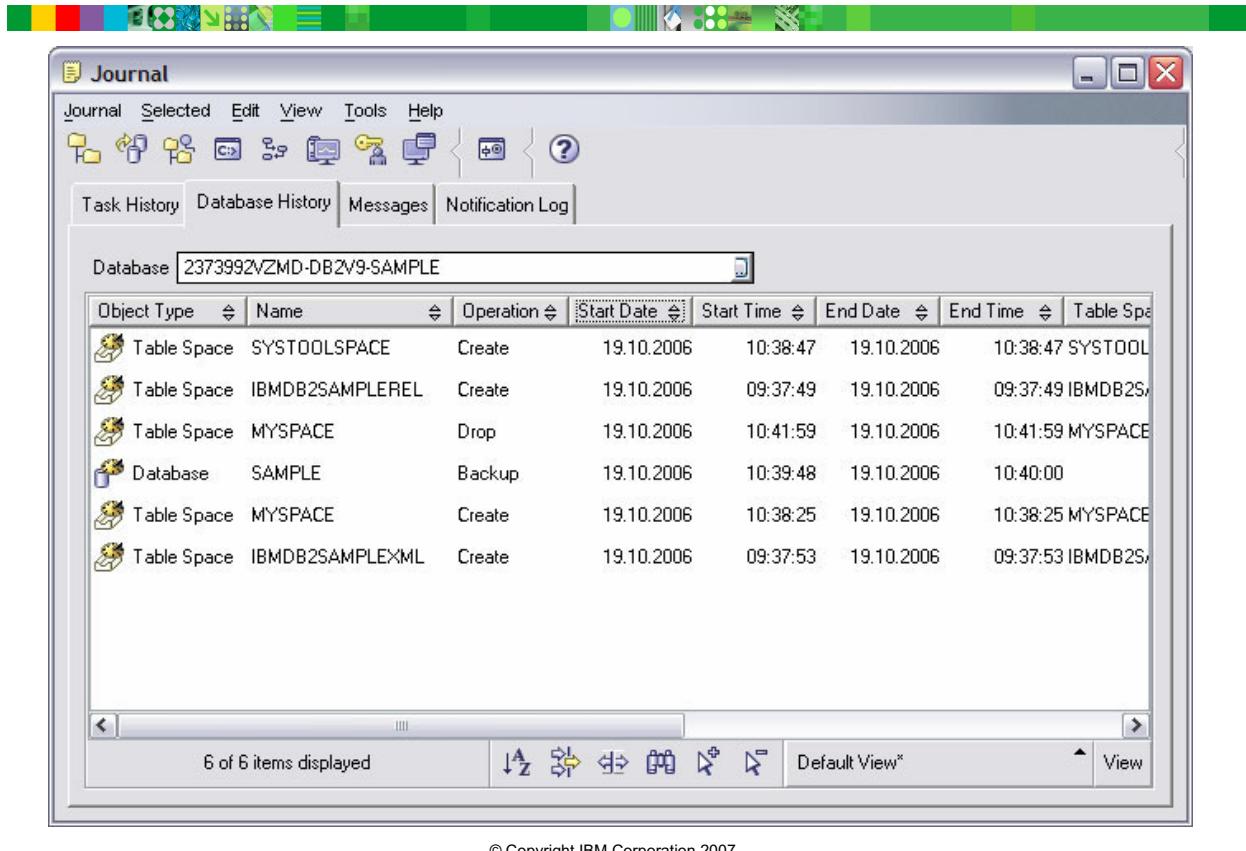


Figure 2-18. Journal

CF238.3

Notes:

The Journal displays historical information about tasks, database actions and operations, Control Center actions, messages, and alerts.

To see the most recent information, click the Refresh button.

The **Task History** page shows the results of tasks that were previously executed. You can use this information to estimate how long future tasks will run.

The Task History page contains one row for each execution of a task. The Task Center contains only one row for each task no matter how many times the task is executed. Each row in the Task Center could be directly related to multiple rows in the Task History page of the Journal.

For each completed execution of a task, you can perform the following actions:

- View the execution results
- View the task that was executed
- Edit the task that was executed
- View the task execution statistics

- Remove the task execution object from the Journal

To perform one of these actions, right-click a completed task execution and select the corresponding action from the pop-up menu that appears.

The **Database History** page shows information from the recovery history file. This file is updated when various operations are performed, including:

- Backup
- Restore
- Roll forward
- Load
- Reorg

This information could be useful if you need to restore a database or table space.

The **Messages** page shows messages that were previously issued from the Control Center and other GUI tools.

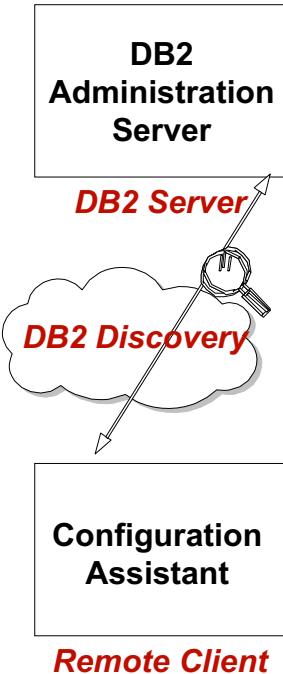
The **Notification Log** page shows information from the administration notification log. You can filter the list, for example to show only alerts from the health monitor.

Several customizable saved views are available via the pull-up menu at the bottom of the Journal.

Configuration Assistant (1 of 2)



- Used to configure client connections
 - Clients can be configured:
 - Manually
 - Automatically using DB2 Discovery
 - Using imported profiles
 - Search the network
- DB2 CA can also:
 - Connect to databases, both LAN and DRDA
 - Perform CLI/ODBC administration
 - Bind applications to database



© Copyright IBM Corporation 2007

Figure 2-19. Configuration Assistant (1 of 2)

CF238.3

Notes:

The Configuration Assistant (CA) is the GUI tool that is used to configure access to remote databases. It can be invoked from the Control Center, DB2 Desktop folder, or from the command line with the db2ca command.

Use the Configuration Assistant to configure your clients. You can also use it as a lightweight alternative to the Control Center, in situations where you do not want to install the complete set of GUI tools.

CA can also be used to change DB2 registry parameters and DB2 instance parameters but, not all available parameters are shown in the lists.

You must configure your DB2 clients so that they can work with the available objects, as follows:

- To access an instance or database on another server or system, DB2 must catalog that system in the node directory of the client.
- To access a database, DB2 needs information about the node where the database resides.

- Each database that will be accessed must be configured at the DB2 client.

From the Configuration Assistant, you can work with existing database objects, add new ones, bind applications, set database manager configuration parameters, and import and export configuration information. The graphical interface makes these complex tasks easier through:

- Wizards that help you perform certain tasks
- Dynamic fields that are activated based on your input choices
- Hints that help you make configuration decisions
- Discovery that can retrieve information about selected database objects

The Configuration Assistant displays a list of the databases to which your applications can connect. Each database is identified by its database alias. You can use the Add Database wizard to add databases to the list. You can use the Change Database wizard to alter the information that is associated with databases in the list.

From the View menu, you can select an advanced view, which uses a notebook to organize connection information by object: Systems, Instance Nodes, Databases, Database Connection Services (DCS), and Data Sources.

You can use the notebook pages to perform object-specific actions.

Configuration Assistant (2 of 2)

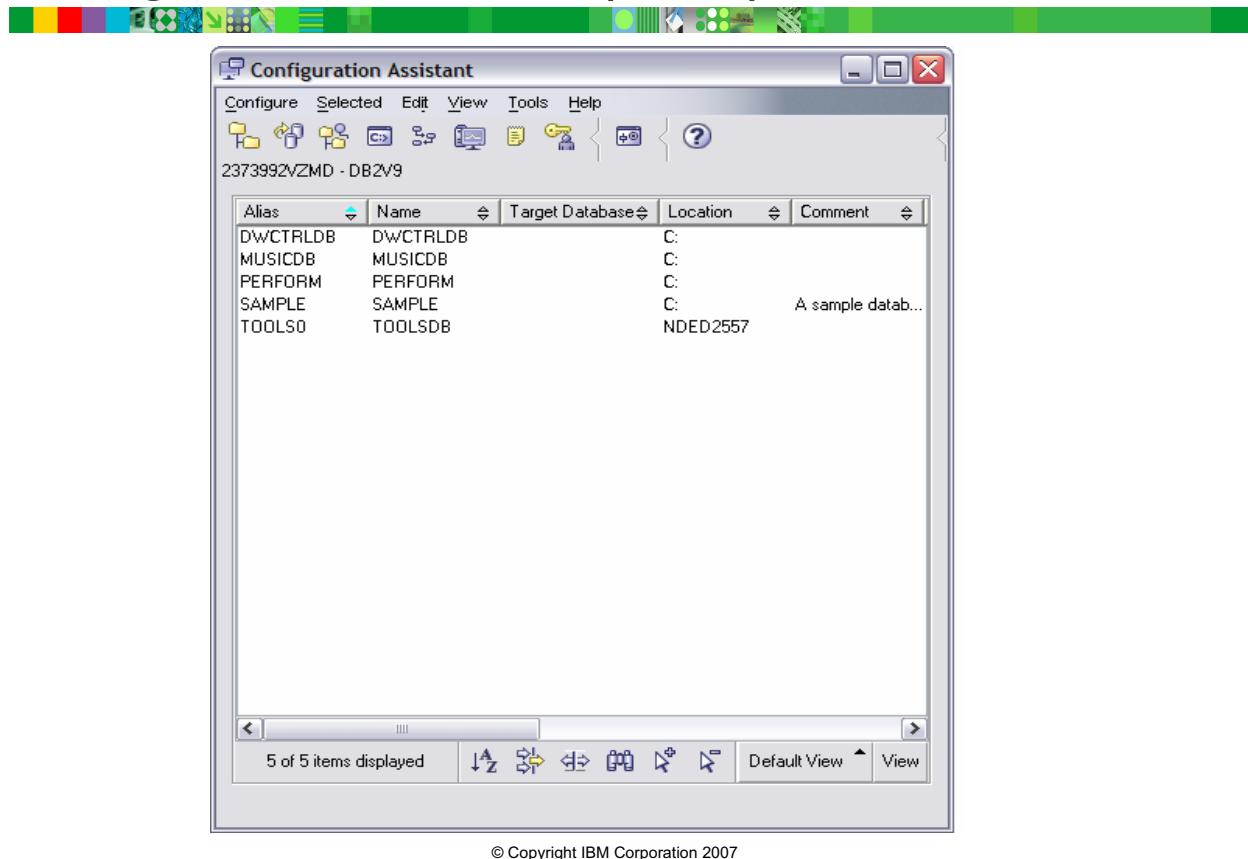


Figure 2-20. Configuration Assistant (2 of 2)

CF238.3

Notes:

The graphic shows an example of using the Configuration Assistant (CA). To configure a client connection click Selected, Add Database Using Wizard, and the Add Database Wizard appears. On this page, you indicate how you will be adding the database to which you want to connect. Each method involves a slightly different set of wizard pages.

- **Use a profile** invokes:
 - Select a database from a profile.
 - Specify an alias for the database.
 - Register this database as a data source.
- **Search the network** invokes:
 - Select a database from the network search result.
 - Specify an alias for the database.
 - Register this database as a data source.

- **Manually configure a connection to a database** invokes:
 - Specify catalog options. (Only appears if the Lightweight Directory Access Protocol (LDAP) is enabled.)
 - Select a communications protocol.
 - Specify communication parameters. (A page tailored to the protocol specified on the previous page.)
 - Specify information for a database on this system. (Only appears if the database is local.)
 - Specify information for a database on a remote system. (Only appears if the database is remote.)
 - Register this database as a data source.
 - Specify the node options.
 - Specify the system options.
 - Specify the security options.
 - Specify the DCS options when adding a DCS entry. (Only appears if TCP/IP or APPN is the communications protocol.)

In order to use the discovery facility to search the network for remote databases, you must do the following:

- Ensure that protocol stacks on the client and server are preinstalled and configured so that they are fully functional at discovery time.
- On every server where a database exists which you want the option of discovering:
 - The DB2 Administration Server service must be installed, configured, and active on the server workstation to support one or more protocols.
 - The database manager configuration parameter of DISCOVER must be set to either KNOWN or SEARCH. DISCOVER is only used when the system is acting as a client.
 - The parameter DISCOVER_INST is used to specify whether this instance can be detected by DB2 discovery. The default, enable, specifies that the instance can be detected, while disable prevents the instance from being discovered.
 - The database parameter of DISCOVER must be enabled for each database to be discovered.
- The code that enables discovery on the client must be installed on the client workstation to support one or more protocols.

Tool settings



- General
 - Option to disable hover help, infopops
 - Option to perform **db2start** at tool startup time
 - Define statement termination character
 - Filtering based on number of rows
 - Maximum size for Sample Contents, Command Center
- Fonts
 - Specify font, size, and color for Menu and Text
- OS/390 and z/OS
 - Specify to use system catalog column names
 - Specify Utility execution options
- Health Center Status Beacons
 - Option to specify to notify through pop-up message, status line, or both
- Scheduler Settings
 - Define default scheduling scheme
 - Define Tools Catalog

© Copyright IBM Corporation 2007

Figure 2-21. Tool settings

CF238.3

Notes:

Using Tool Settings, preferences for the Control Center can be changed. From the Control Center toolbar, click **Tool Settings** icon or choose **Tool Settings** from the *Tools* menu bar option.

This opens the Tool Settings notebook where you can change settings for:

- **General** — Enable or disable hover help, whether to automatically display infopops, whether to start (local) DB2 automatically on tools startup or not, define the statement terminator for SQL statements, and to set the filtering based on number of rows. You can also set the maximum size of rows in Sample Contents and Command Editor.
- **Fonts** — Change the font in which text and menus appear in the DB2 user interface.
- **OS/390 and z/OS** — Identify whether to use system catalog column names as column headings. Specify utility execution options: whether to edit options each time a utility runs, specify the online execution utility ID template, specify if execution should continue if an error is encountered, specify if grouping of objects should occur for parallel utility execution, indicate the maximum number of objects to process in parallel

for online execution, indicate the maximum number of jobs to run in parallel for batch execution, and indicate the maximum number of objects per batch job.

- **Health Center Status Beacons** — Indicate if notification should take place through a popup message or through the status line.
- **Scheduler Settings** — Identify if the default scheduling scheme should be server scheduling or centralized scheduling. Define the tools catalog.

After modifying the settings, close the Tool Settings window and most changes will take place immediately. Some changes, such as those made to fonts, will not take effect until the Control Center is restarted.

2.3 Overview of the Administration Server

DB2 Administration Server



- DB2 control point to assist with server tasks
- Must have running DAS on server to use Configuration Assistant, Control Center, Task Center, or Development Center
- Might be created and configured at installation time
- Assists with:
 - Enabling remote administration of DB2 servers
 - Providing the facility for job management, including scheduling
 - Providing a means for discovering information about the configuration of DB2 instances, databases, and other DB2 Administration Servers in conjunction with DB2 Discovery

© Copyright IBM Corporation 2007

Figure 2-22. DB2 Administration Server

CF238.3

Notes:

The DB2 Administration Server (DAS) is a control point used only to assist with tasks on DB2 servers. You must have a running DAS if you want to use available tools like the Configuration Assistant, the Control Center, the Task Center, or the Development Center. DAS assists the Control Center and Configuration Assistant when working on the following administration tasks:

- Enabling remote administration of DB2 servers.
- Providing the facility for job management, including the ability to schedule the running of both DB2 and operating system command scripts. These command scripts are user-defined.
- Defining the scheduling of jobs, viewing the results of completed jobs, and performing other administrative tasks against jobs located either remotely or locally to the DAS using the Task Center.
- Providing a means for discovering information about the configuration of DB2 instances, databases, and other DB2 administration servers in conjunction with the DB2 Discovery utility. This information is used by the Configuration Assistant and the Control

Center to simplify and automate the configuration of client connections to DB2 databases.

You can only have one DAS on a machine. DAS is configured during installation to start when the operating system is booted.

DAS is used to perform remote tasks on the server system and the host system on behalf of a client request from the Control Center, the Configuration Assistant, or any of the other available tools.

The DAS on Windows and Linux/UNIX includes a scheduler to run tasks (such as DB2 and operating system command scripts) defined using the Task Center. Task information such as the commands to be run; schedule, notification, and completion actions associated with the task, and run results are stored in a DB2 database called the Tools Catalog database. The Tools Catalog database is created as part of the setup. It can also be created and activated through the Control Center, or through the CLP using the CREATE TOOLS CATALOG command.

Using the Administration Server

- Creating the Administration Server
 - `dascrt ASName` (UNIX)
 - `db2admin create` (Windows)
- Starting and stopping the Administration Server
 - `db2admin start`
 - `db2admin stop`
- Listing the Administration Server
 - `db2admin`
- Configuring the Administration Server
 - `db2 get admin cfg`
 - `db2 update admin cfg using ...`
- Removing the Administration Server
 - `dasdrop ASName` (UNIX)
 - `db2admin drop` (Windows)

© Copyright IBM Corporation 2007

Figure 2-23. Using the Administration Server

CF238.3

Notes:

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

On a Linux/UNIX system, to create an Administration Server, you must have root authority to execute the `dascrt` command. The syntax is as follows:

```
dascrt ASName      (Linux/UNIX)  
db2admin create    (Windows)
```

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

- Following a system reboot after installation
- Using a manual start (`db2admin start`)

To manually start or stop the DAS on Windows, you must first log on to the machine using an account or user ID that belongs to either Administrators, Server Operators, or Power Users groups. To manually start or stop the DAS on Linux/UNIX, the account or user ID must be made part of the *dasadm_group*. The *dasadm_group* is specified in the DAS configuration parameters. You can access the DAS configuration parameters by issuing:

```
db2 get admin cfg  
db2 update admin cfg using ...
```

To start or stop the DAS, issue:

```
db2admin start  
db2admin stop
```

The Administration Server is automatically started after each system reboot.

To obtain the name of the Administration Server on your system, use the `db2admin` command:

```
db2admin
```

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

```
db2admin stop
```

If you wish to preserve the files stored for the DAS, you must first back up the files:

- On Windows, in the **db2das00** subdirectory under the **sqllib** subdirectory.
- On Linux/UNIX, in the **das** subdirectory under the **home** directory of the DAS.

On Windows, issue:

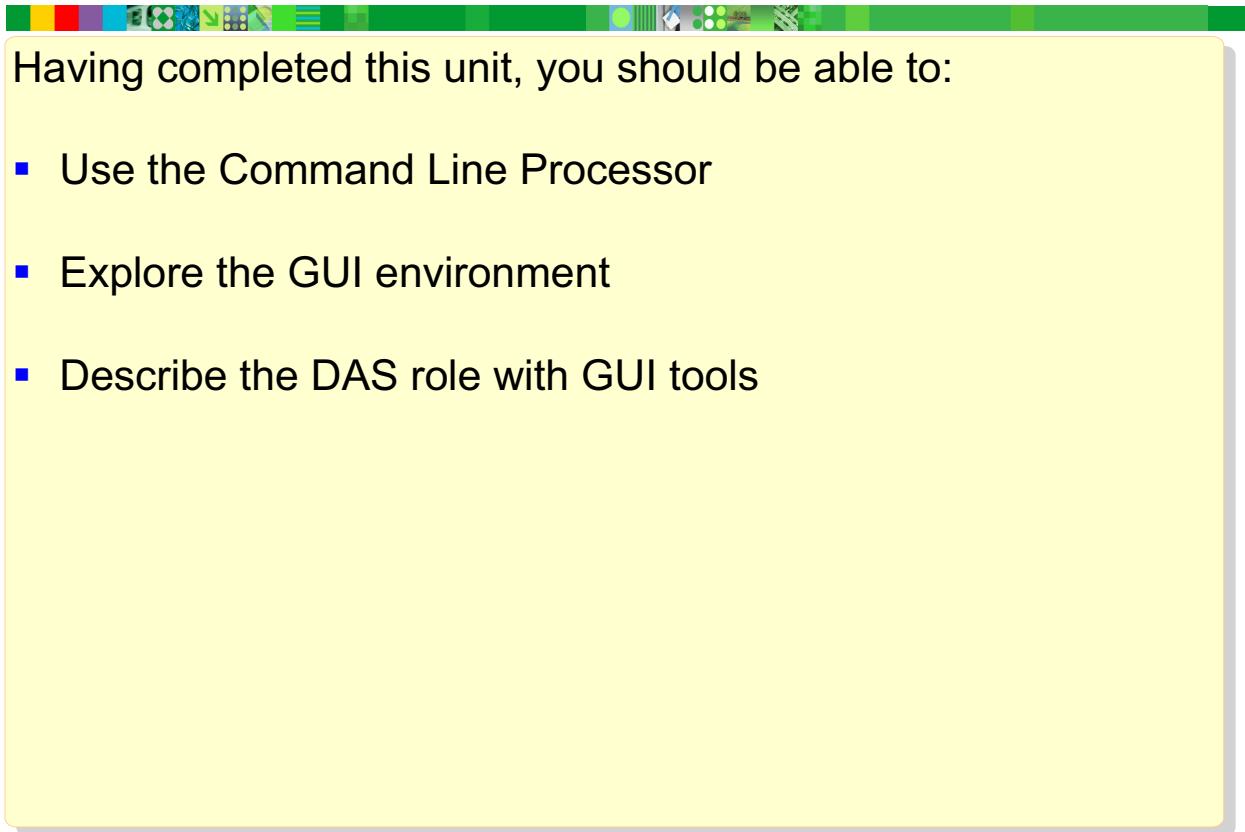
```
db2admin drop
```

On Linux/UNIX, log out as the Administration Server owner. Then log in as root and remove the Administration Server instance using the `dasdrop` command as follows:

```
dasdrop ASName
```

where ASName is the name of the instance being removed. The `dasdrop` command removes the `sql1ib` directory under the `home` directory of the Administration Server.

Unit summary



© Copyright IBM Corporation 2007

Figure 2-24. Unit summary

CF238.3

Notes:

Unit 3. The DB2 environment

What this unit is about

This unit describes the environment, including how to create and drop a database server system (Instance). The ability to start and stop the server system will be explained, and the configuration mechanism and parameters for the database system will also be covered.

What you should be able to do

After completing this unit, you should be able to:

- Specify the key features of an Instance
- Create and drop an Instance
- Use **db2start** and **db2stop**
- Distinguish between types of configuration
- Describe and modify the Database Manager Configuration

How you will check your progress

Accountability:

- Machine exercises

References

Administration Guide: Planning

Administration Guide: Implementation

Command Reference

<http://www.ibm.com/software/data/db2/udb/support/manualsv9.html>
DB2 V9 Manuals

Unit objectives



After completing this unit, you should be able to:

- Specify the key features of an instance
- Create and drop an instance
- Use **db2start** and **db2stop**
- Distinguish between types of configuration
- Describe and modify the Database Manager Configuration

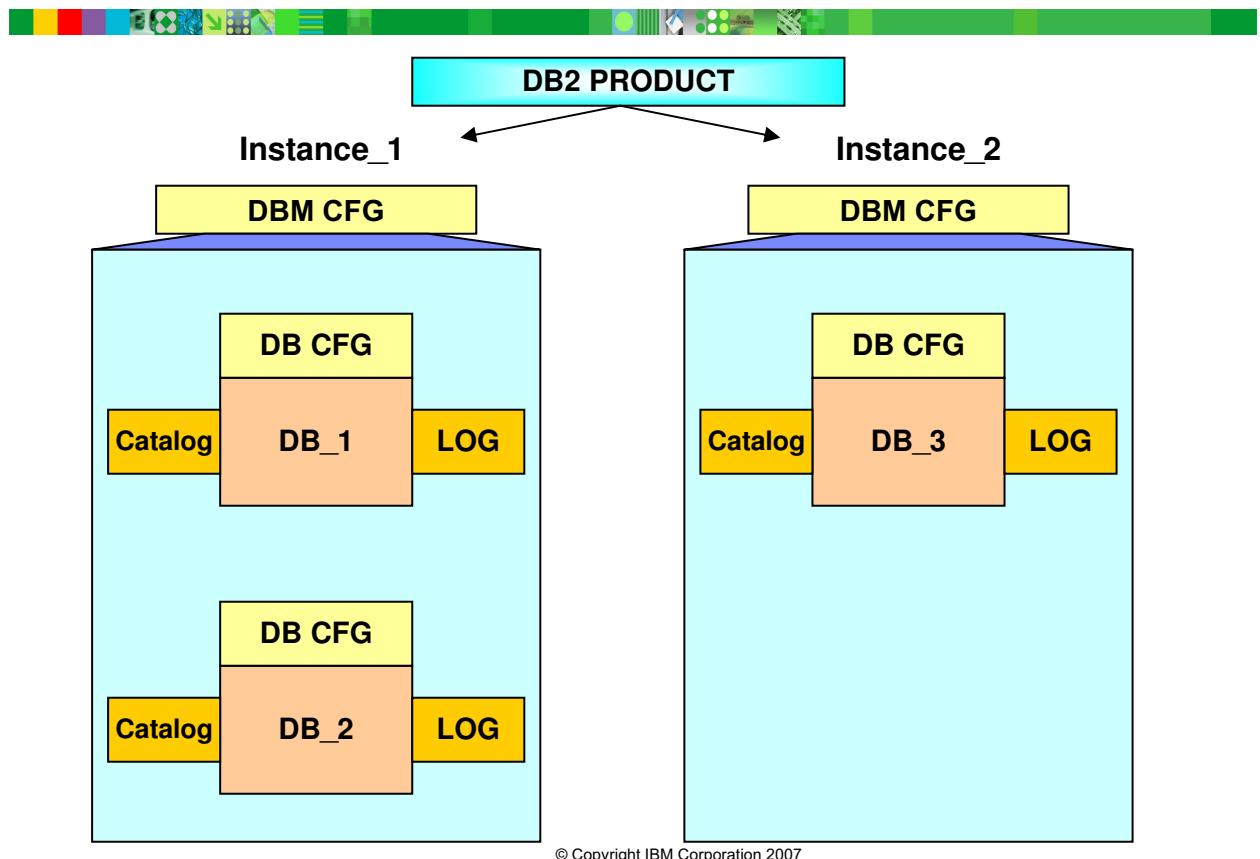
© Copyright IBM Corporation 2007

Figure 3-1. Unit objectives

CF238.3

Notes:

What is an instance?



© Copyright IBM Corporation 2007

Figure 3-2. What is an Instance?

CF238.3

Notes:

Multiple instances of the database manager may be created on a single workstation. This means that you can create several instances of the same product on a physical machine, and have them running concurrently.

DB2 configuration files contain parameter values that define the resources allocated to DB2 and individual databases. There are two types of configuration files: the *database manager configuration file*, for the DB2 instance as a whole, and the *database configuration file*, for each individual database.

The *database manager configuration file* is created when an *instance* of DB2 is created, and affects the system resources allocated to DB2 for an individual instance. Its parameters have global applicability, independent of any one database stored in the system.

A *database configuration file* for an individual database is created when the database is created. There is one configuration file per database. The parameters in this configuration file specify the amount of resources to be allocated to that database. Many of these

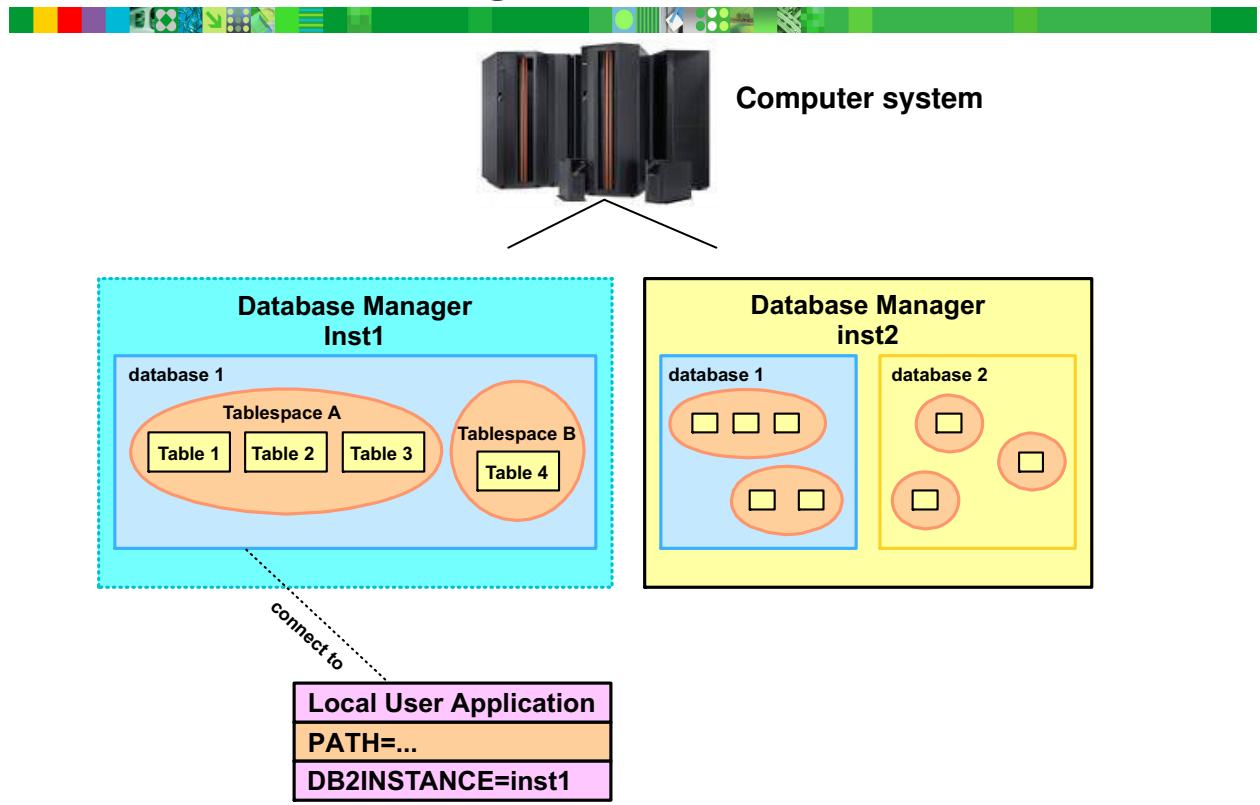
parameters can be changed to improve the performance or increase capacity. Different changes may be required depending on the type of activity in that specific database.

DB2 creates and maintains a set of system catalog tables for EACH database. These tables contain information such as descriptions of tables, views, and packages.

On Windows and Linux/UNIX-based platforms, each *instance* is a virtual copy of the code using symbolic links to the physical code. This allows a separate physical copy of a new version of code to be installed and instances to be created from the new code.

DB2 users have the power of distributed query across any DB2 family database or OLE DB source. This means that users and applications can use the DB2 SQL syntax and APIs to access data that resides at heterogeneous data sources. With this functionality, users and applications have the capability of referencing multiple data sources in a single SQL statement.

The Database Manager instance



© Copyright IBM Corporation 2007

Figure 3-3. The Database Manager instance

CF238.3

Notes:

Each node can have several instances.

Each instance can manage several databases. However, a given database belongs to only one instance.

A database is a collection of tables.

The tables are logically grouped in table spaces.

Access to a particular database or instance will be determined by the PATH and DB2INSTANCE environment variables or through an CONNECT or ATTACH command in the application.

Create and drop an instance

- CREATE (different on Linux/UNIX and Windows)
 - Prerequisites met?
 - Creates the Instance
 - Creates the SQLLIB subdirectory
 - Creates needed directories in the SQLLIB subdirectory
 - Creates the DBM CFG with defaults
- DROP
 - Instance must be stopped and no applications are allowed to be connected to the databases in this instance
 - Does not remove databases (catalog entries)
 - Removes the Instance

© Copyright IBM Corporation 2007

Figure 3-4. Create and drop an instance

CF238.3

Notes:

To create an Instance, we must distinguish between the Operating Systems Linux/UNIX and Windows.

Linux/UNIX environments:

- We must take care from which path we issue the **db2icrt** command (because of possible different installations and fixpacks). Furthermore we need an Instance-Owner (User) and also we need another user, to run fenced processes like UDFs and Stored Procedures.
- Because the authorities of an instance need a Group and not a User, we additionally need to create two groups: one for the administrator and instance owner and one for the fenced user.
- During the instance creation, there will be created different directories and subdirectories for configuration and information in the \$HOME directory of the instance owner:
 - /sqllib/

- adm
- cfg
- ctrl
- db2cshrc
- db2dump
- db2nodes.cfg
- db2profile
- db2systm
- security
- sqldbdrr
- and so forth. Furthermore, there will be a lot of *links* to the original installation files of DB2.

Windows environments:

- There is no need to create special users or groups, but it is recommended to do it for security reasons (SYSADM).
- The instance creation creates a directory in the root directory for example, **c:\IBM\SQLLIB\InstanceName**. Here all other subdirectories and files for configuration and information can be found:
 - ctrl
 - db2systm
 - log
 - security
 - TMP

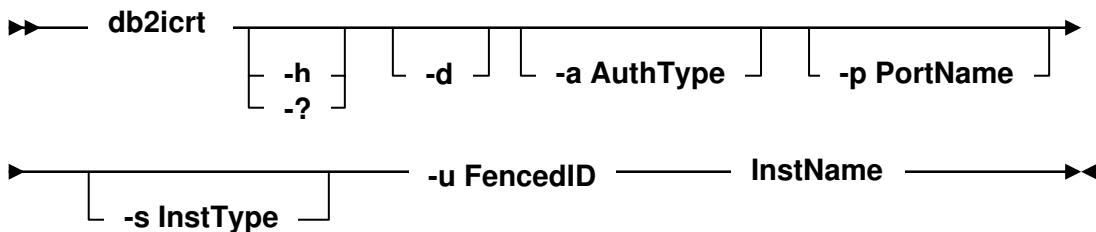
Before dropping an instance, you must:

- If needed, backup all databases or if not needed, drop those databases. Dropping an instance does not remove the databases (uncatalog).
- Force all applications connected to any database in the instance.
- Stop the instance.
- All relevant directories and subdirectories will be removed (please verify).

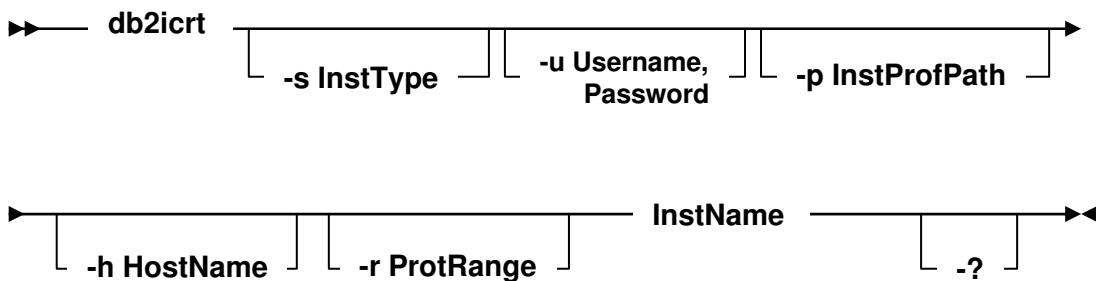
db2icrt command



Linux/UNIX



Windows



© Copyright IBM Corporation 2007

Figure 3-5. db2icrt command

CF238.3

Notes:

On Linux and UNIX-based systems, this utility is located in the ***DB2DIR/instance*** directory, where DB2DIR represents the installation location where the current version of the DB2 database system is installed. On Windows operating systems, this utility is located under the ***DB2PATH\bin*** directory where DB2PATH is the location where the DB2 copy is installed.

You must have Root access on Linux and UNIX-based systems or Local Administrator authority on Windows operating systems.

Command parameters for Linux and UNIX operating systems:

- **-h or -?** — Displays the usage information.
- **-d** — Turns debug mode on. Use this option only when instructed by DB2 Support.
- **-a *AuthType*** — Specifies the authentication type (SERVER, CLIENT or SERVER_ENCRYPT) for the instance. The default is SERVER.
- **-p *PortName*** — Specifies the port name or number used by the instance. This option does not apply to client instances.

- **-s InstType**— Specifies the type of instance to create. Use the **-s** option only when you are creating an instance other than the default for your system. Valid values are:
 - **client**— Used to create an instance for a client. Use this value if you are using DB2 Connect Personal Edition.
 - **ese**— Used to create an instance for a database server with local and remote clients.
 - **wse**— Used to create an instance for DB2 Workgroup Server Edition, DB2 Express Edition and DB2 Connect Enterprise Edition.
- **-u Fenced ID**— Specifies the name of the user ID under which fenced user-defined functions and fenced stored procedures will run. The **-u** option is required if you are not creating a client instance.
- **InstName**— Specifies the name of the instance which is also the name of an existing user in the operating system.

Command parameters for Windows operating systems:

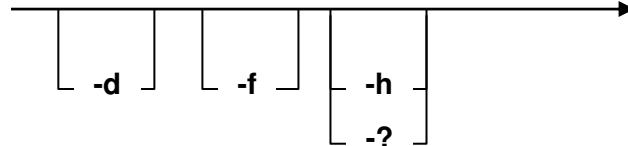
- **InstName**— Specifies the name of the instance.
- **-s InstType**— Specifies the type of instance to create. Valid values are:
 - **client**— Used to create an instance for a client. Use this value if you are using DB2 Connect Personal Edition.
 - **standalone**— Used to create an instance for a database server with local clients.
 - **ese**— Used to create an instance for a database server with local and remote clients.
 - **wse**— Used to create an instance for DB2 Workgroup Server Edition, DB2 Express Edition and DB2 Connect Enterprise Edition.
- **-u Username, Password**— Specifies the account name and password for the DB2 service. This option is required when creating a partitioned database instance.
- **-p InstProfPath**— Specifies the instance profile path.
- **-h HostName**— Overrides the default TCP/IP host name if there is more than one for the current machine. The TCP/IP host name is used when creating the default database partition (database partition 0). This option is only valid for partitioned database instances.
- **-r PortRange**— Specifies a range of TCP/IP ports to be used by the partitioned database instance when running in MPP mode. For example, -r 50000,50007. The services file of the local machine will be updated with the following entries if this option is specified:
 - DB2_InstName_baseport/tcp
 - DB2_InstName_END_endport/tcp
- **-?**— Displays usage information.

db2idrop command



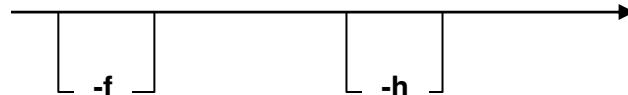
Linux/UNIX

►► db2idrop – InstName



Windows

►► db2idrop – InstName



© Copyright IBM Corporation 2007

Figure 3-6. db2idrop command

CF238.3

Notes:

Removes a DB2 instance that was created by db2icrt. You can only drop instances that are listed by db2ilist for the same DB2 copy where you are issuing db2idrop from.

On Linux and UNIX-based systems, this utility is located in the **DB2DIR/instance** directory, where DB2DIR represents the installation location where the current version of the DB2 database system is installed. On Windows operating systems, this utility is located under the **DB2PATH\bin** directory where DB2PATH is the location where the DB2 copy is installed.

You must have Root access on Linux and UNIX-based systems or Local Administrator on Windows operating systems.

Command parameters for Linux and UNIX-based systems:

InstName — Specifies the name of the instance.

- **-d** — Enters debug mode, for use by DB2 Service.
- **-f** — Specifies the force applications flag. If this flag is specified, all the applications using the instance will be forced to terminate.

- **-h or -?** — Displays the usage information.

Command parameters for Windows-based systems:

- **InstName** — Specifies the name of the instance.
- **-f** — Specifies the force applications flag. If this flag is specified, all the applications using the instance will be forced to terminate.
- **-h** — Displays usage information.

Starting and stopping an instance



db2start



db2stop

© Copyright IBM Corporation 2007

Figure 3-7. Starting and stopping an instance

CF238.3

Notes:

db2start

Starts the current database manager instance background processes on a single database partition or on all the database partitions defined in a partitioned database environment. Start DB2 at the server before connecting to a database, precompiling an application, or binding a package to a database. **db2start** can be executed as a system command or a CLP command.

The **db2start** command launches the DB2 product installation as a Windows service. The DB2 product installation on Windows can still be run as a process by specifying the **/D** switch when invoking **db2start**. The DB2 product installation can also be started as a service using the Control Panel or the **NET START** command.

Since **db2start** launches a Windows service, you must meet Windows requirements for starting a service. If Extended Security is disabled, you must be a member of the Administrators, Server Operators or Power Users group. If Extended Security is enabled, you must be a member of either the Administrators group or the DB2ADMNS group to start the database.

If a **db2start** operation in a multi-partition database is not completed within the value specified by the *start_stop_timeout* database manager configuration parameter, the database partitions that have timed out will be killed internally (all resources associated with the database partition will be removed). Environments with many database partitions with a low value for *start_stop_timeout* might experience this behavior. To resolve this behavior, increase the value of *start_stop_timeout*.

Alternatively, you can also issue a *start dbm* or *start database manager* command.

db2stop

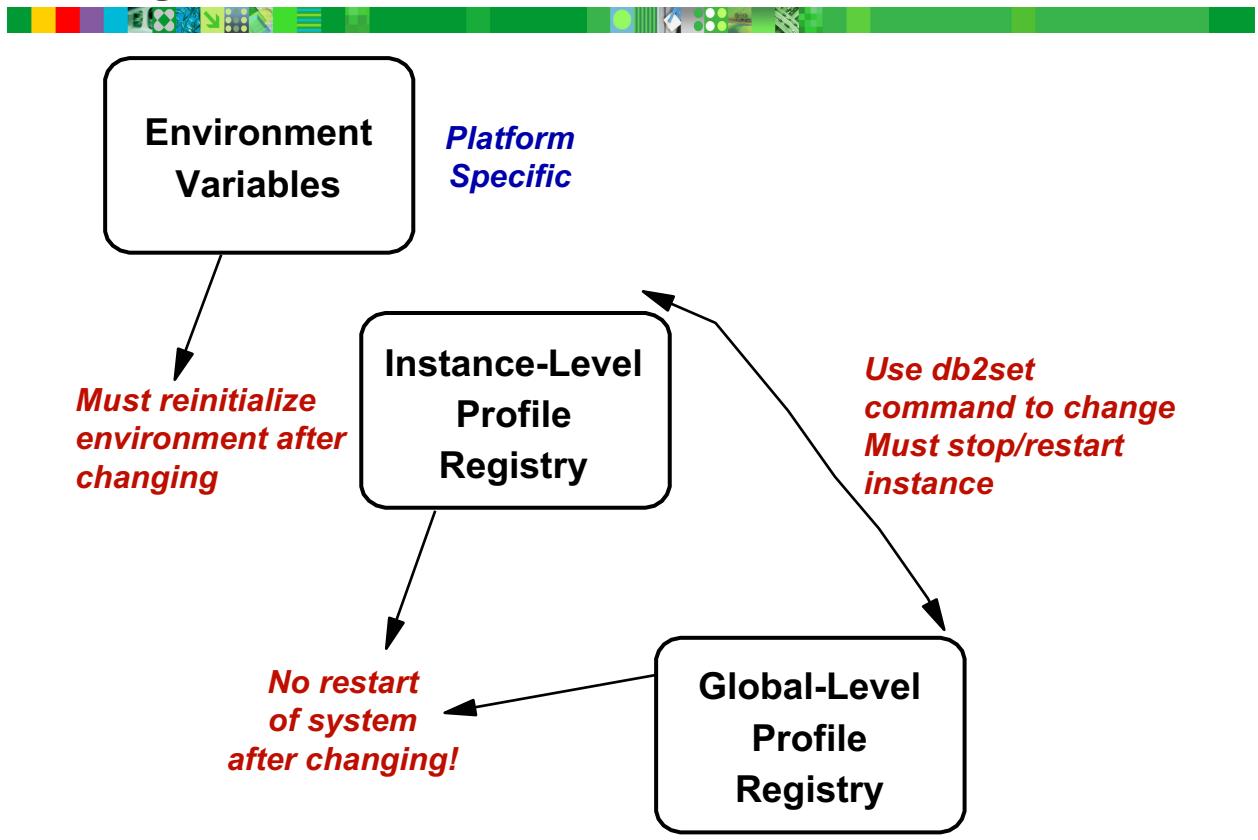
Stops the current database manager instance. **db2stop** can be executed as a system command or a CLP command.

If a **db2stop** operation in a multi-partition database is not completed within the value specified by the *start_stop_timeout* database manager configuration parameter, the database partitions that have timed out will be killed internally (all resources associated with the database partition will be removed). Environments with many database partitions with a low value for *start_stop_timeout* might experience this behavior. To resolve this behavior, increase the value of *start_stop_timeout*.

If a db2stop command is issued and there are active databases or connections to the databases in the instance, an error is returned. If you need to force the stop (so applications will be forced, not committed work rolled back) you can issue **db2stop force**.

Alternatively, you can also issue a *stop dbm* or *stop database manager* command.

Setting DB2 variable values



© Copyright IBM Corporation 2007

Figure 3-8. Types of configuration

CF238.3

Notes:

There are different types of configurations related to the Instance:

- Registry settings like DB2PATH, DB2COMM, and so forth. These settings can be changed using the *db2set* command.
- Configuration settings in the Database Manager Configuration file. These settings can be changed using the GUI in the Control Center, selecting the instance, right-clicking and selecting Configure parameters, or with the command *update dbm cfg using ParameterName ParameterValue*.
- Some of the DBM CFG parameters can be changed online, which means the change takes effect immediately. Some of them can be changed, but the change will be effective after an instance restart. Using the GUI, you might see which of them are updatable online and which offline.
- There are also several parameters which are or can be set to AUTOMATIC so you don't have to care about the values.

The db2set command

- Command line tool for DB2 Profile Registry administration
- Displays, sets, resets, or removes profile variables

```
db2set variable=value
-g
-e
-i instance [db-partition-number]
-n DAS node [[-u user id] [-p password]]
-u
-ul
-ur
-p
-r
-l
-lr
-v
-? (or -h)
-all
```

© Copyright IBM Corporation 2007

Figure 3-9. Database Manager configuration

CF238.3

Notes:

You can see the database manager configuration in different ways. The DBM configuration GUI can be accessed via the CC (Control Center) and CA (Configuration Assistant). Not all possible parameters are listed, but the command line can be used to change others.

- Using the CLP or Command Editor and then issuing *db2 get dbm cfg (show detail)*.
- Using the GUI from the Control Center: Select the instance, right-click with the mouse and select Configure parameters. This method has advantages:
 - The parameters are grouped
 - You can see if they could be changed online or offline
 - You will get a hint (short description) if you highlight a specific parameter

The DBM CFG parameter groups are the following:

- Administration related parameters
- Application related parameters

- Communication related parameters
- Diagnostic related parameters
- Environment related parameters
- Miscellaneous related parameters
- Monitoring related parameters
- Parallel functionality related parameters
- Performance related parameters

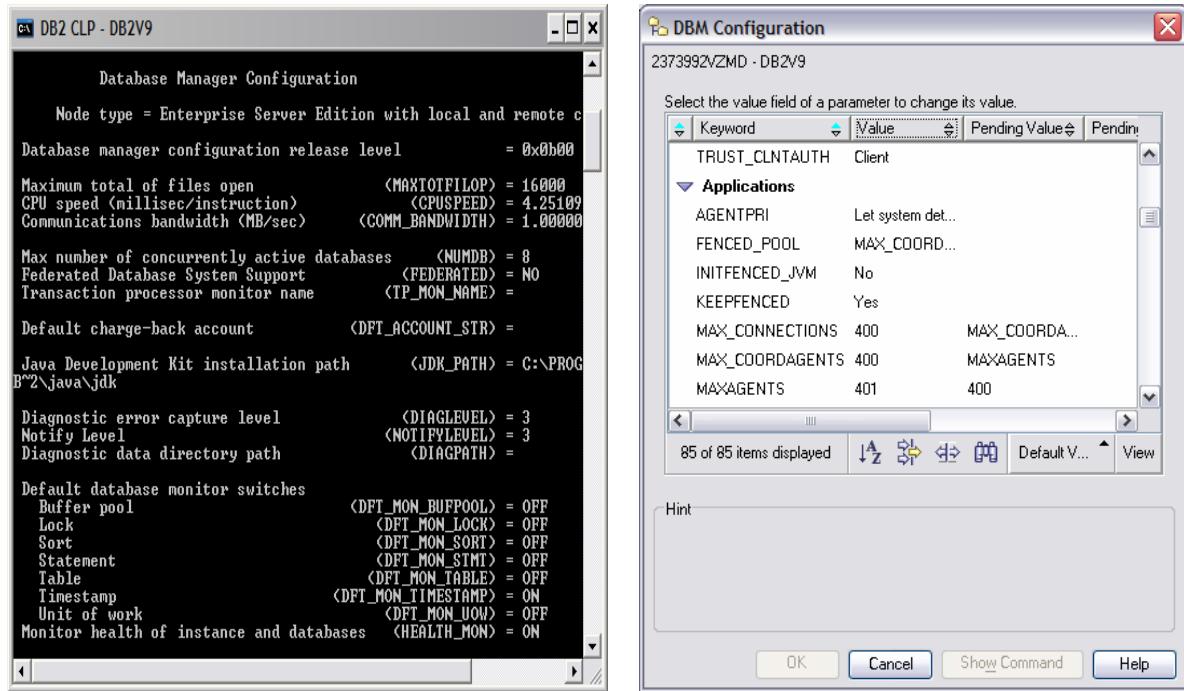
To change the value of any of those parameters, you can:

Use the CLP or Command Editor with the command *update dbm cfg using ParameterName ParameterValue*

Use the GUI from the Control Center: Select the instance, right-click with the mouse and select Configure parameters. Here you might also have some advantages:

- You will get a hint (short description) if you highlight a specific parameter
- When you click to the VALUE field you will see:
 - Actual value
 - Possible range of values
 - If this parameter could be changed online
 - Hint (short description) of this parameter

Database Manager configuration



© Copyright IBM Corporation 2007

Figure 3-10. Unit summary

CF238.3

Notes:

Unit 4. Creating databases and data placement

What this unit is about

When you are ready to implement your database, there are a number of factors which should be considered about the physical environment in which the database will be implemented. These factors include choosing an instance in which you will create your database, how much space will be required to store your data, where data should be physically located, and what kind of table space should be used for storing data. This unit will address considerations in these areas.

What you should be able to do

After completing this unit, you should be able to:

- Review specifics of creating a database
- Explore the System Catalog tables and views
- Compare DMS versus SMS table spaces
- Describe how to setup and manage a DB2 database with Automatic Storage enabled
- Differentiate between table spaces, containers, extents, and pages
- Define table spaces
- Use the get snapshot for tablespaces command to display table space statistics
- Explore Database configuration parameters

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

Administration Guide

Unit objectives



After completing this unit, you should be able to:

- Review specifics of creating a database
- Explore the System Catalog tables and views
- Compare DMS versus SMS table spaces
- Describe how to setup and manage a DB2 database with Automatic Storage enabled
- Differentiate between table spaces, containers, extents, and pages
- Define table spaces
- Use the **get snapshot for tablespaces** to display table space statistics
- Explore database configuration parameters

© Copyright IBM Corporation 2007

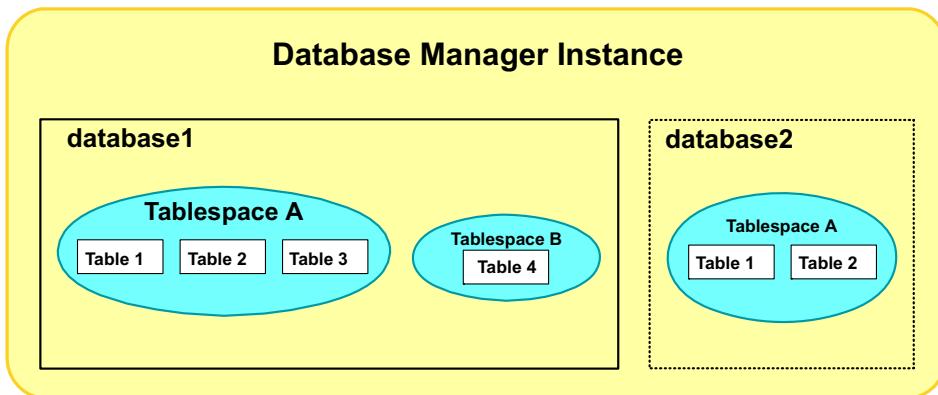
Figure 4-1. Unit objectives

CF238.3

Notes:

4.1 Create database overview

Create database overview



- Databases are created within a Database Manager instance
- Table spaces are a logical layer created within database
- Tables are created within table spaces

© Copyright IBM Corporation 2007

Figure 4-2. Create database overview

CF238.3

Notes:

Each instance can have one or more databases defined in it.

Each database can have three or more table spaces associated with it. The table spaces are a logical level between the database and the tables stored in that database. Table spaces are created within a database, and tables are created within table spaces.

Steps for database creation



1. Create database
2. Create database configuration file and set default values
3. Bind database utilities to the database (db2ubind.lst)
4. Define SYSCATSPACE, TEMPSPACE1, and USERSPACE1 table spaces
5. Set up all of the system catalog tables and allocate the database recovery log
6. Catalog database in local database directory and system database directory
7. Assign codeset, territory, and collating sequence
8. Create SYSCAT, SYSFUN, SYSIBM, SYSSTAT schemata
9. Grant the following privileges to database Creator:
 - DBADM authority with CONNECT, CREATETAB, BINDADD, IMPLICIT_SCHEMA, CREATE_NOT_FENCED_ROUTINE, CREATE_EXTERNAL_ROUTINE, QUIESCE_CONNECT, and LOAD authorities
10. Grant the following privileges to Public (If RESTRICTIVE is not specified):
 - SELECT privilege on system catalog tables and views
 - BIND and EXECUTE privileges to PUBLIC for each successfully bound utility
 - CREATETAB, BINDADD, IMPLICIT_SCHEMA, and CONNECT authorities
 - USE privilege on USERSPACE1 table space

© Copyright IBM Corporation 2007

Figure 4-3. Steps for database creation

CF238.3

Notes:

When you create a database, the database manager:

1. Creates a database in the specified subdirectory.
2. Creates the database configuration file and sets the default values.
3. Binds previously defined database manager bind files to the database (these are listed in db2ubind.lst).
4. Creates SYSCATSPACE, TEMPSPACE1, AND USERSPACE1 table spaces.
5. Sets up all of the system catalog tables needed by the database and allocates the database recovery log.
6. Catalogs the database in the following database directories:
 - Server's local database directory on the path indicated by path or, if the path is not specified, the default database path defined in the database manager system configuration file. A local database directory resides on each operating system file system (UNIX) or drive (Windows) that contains a database.
 - Server's system database directory for the attached instance. The resulting directory entry will contain the database name and a database alias. If the command was

issued from a remote client, the client's system database directory is also updated with the database name and alias.

- Creates a system or a local database directory if neither exists. If specified, the comment and code set values are placed in both directories.
7. Stores the specified codeset, territory, and collating sequence. A flag is set in the database configuration file if the collating sequence consists of unique weights, or if it is the identity sequence.
 8. Creates the SYSCAT, SYSFUN, SYSIBM, and SYSSTAT schemata with SYSIBM as the owner.
 9. Grants the following privileges:
 - DBADM authority with CONNECT, CREATETAB, BINDADD, IMPLICIT_SCHEMA, CREATE_NOT_FENCED_ROUTINE (for user-defined functions and stored procedures), CREATE_EXTERNAL_ROUTINE (for user-defined functions and stored procedures that are implemented using external code), QUIESCE_CONNECT and LOAD authorities to database creator
 - SELECT privileges on each system catalog to PUBLIC
 - BIND and EXECUTE privileges to PUBLIC for each successfully bound utility
 - CREATETAB, BINDADD, IMPLICIT_SCHEMA, and CONNECT authorities to PUBLIC
 - USE privilege on USERSPACE1 table space to PUBLIC

Create Database Wizard

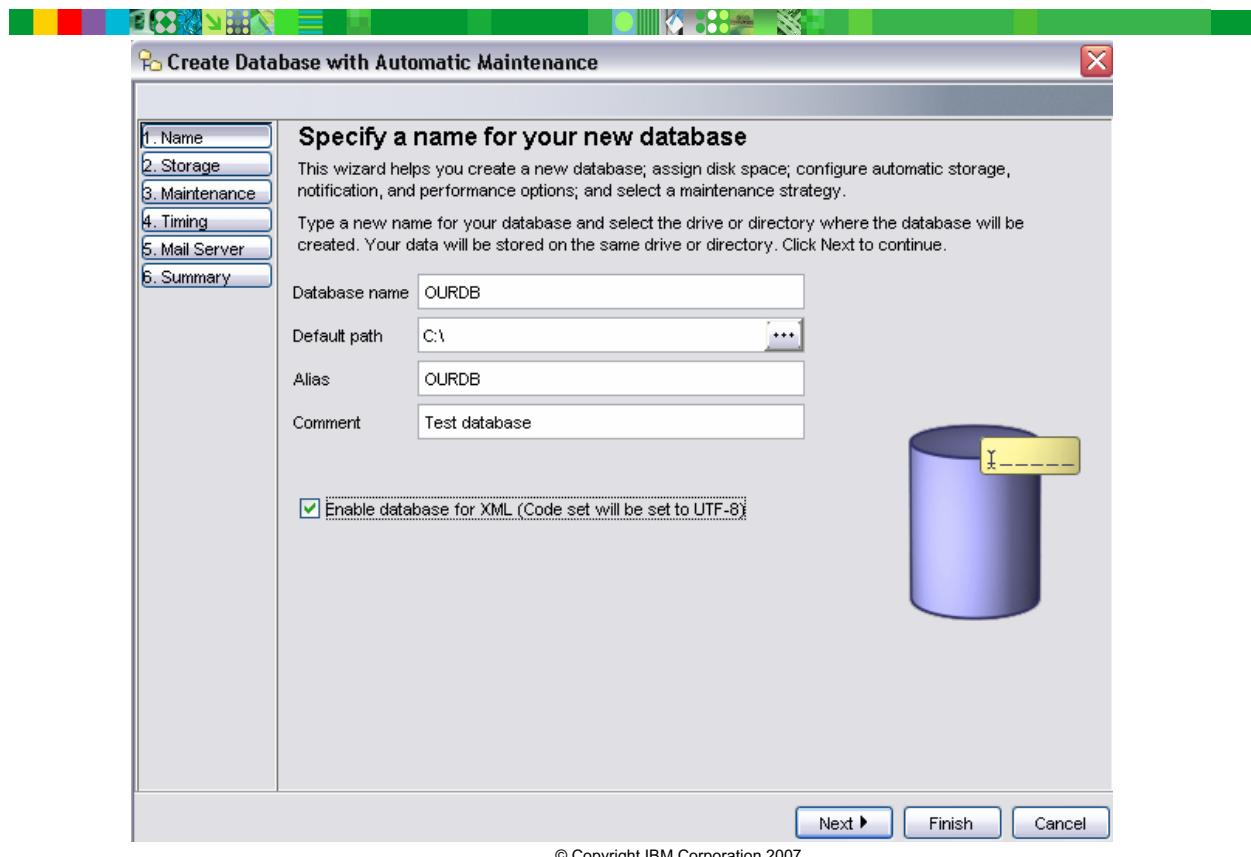


Figure 4-4. Create Database Wizard

CF238.3

Notes:

Only a user with SYSADM or SYSCTRL authority may create databases. A database can be created using CLP, or the Control Center, or through an application program invoking an API. The Create Database Wizard Graphical User Interface or the Create Database Wizard with Automatic Maintenance (GUI) can be displayed through the Control Center.

From the Control Center, expand the object tree until you find the **Databases** folder. Right-click the Databases folder, and then select **Create Database--> Automatic Maintenance** from the pop-up menu. The Create Database wizard opens.

Complete each of the applicable wizard pages. Each page is discussed below. The **Finish** push button is available when you complete enough information for the wizard to create your database. Click **Finish** to create your database.

Name

On this page, you specify a name for your database and select a drive on which to store it. Use the following pages if you want to customize the database. You can also specify whether you want to enable the database for XML.

User Tables

Use this page to select how your user tables should be managed. If you use System Managed Space (SMS), the operating system's file system manager allocates and manages the space where the user tables are stored. The storage model typically consists of many files, representing table objects, stored in the file system space. If you use Database Managed Space (DMS), the database manager controls the storage space. The storage model consists of a limited number of devices, whose space is managed by DB2.

Catalog Tables

Use this page to select how your catalog tables should be managed. If you use System Managed Space (SMS), the operating system's file system manager allocates and manages the space where the catalog tables are stored. If you use Database Managed Space (DMS), the database manager controls the storage space.

Temporary Tables

Use this page to select how your temporary tables should be managed. If you use System Managed Space (SMS), the operating system's file system manager allocates and manages the space where the temporary tables are stored. If you use Database Managed Space (DMS), the database manager controls the storage space.

Performance

This page allows you to set performance parameters for your database.

Region

Use this page to specify the locale of the database in order to determine the character set that the database will use. Also use this page to indicate the collating sequence, which determines how character strings are to be sorted. You cannot change the collating sequence after the database has been created.

Summary

This page allows you to review the decisions that you made for your create database operation, and to view the command that will create your database.

In the **Wizard with Automatic Maintenance**, you can additionally indicate such things as: specifying if you will have an offline maintenance window or not, the start time and day for your maintenance window, and contacts for notification.

Create database syntax (1 of 3)

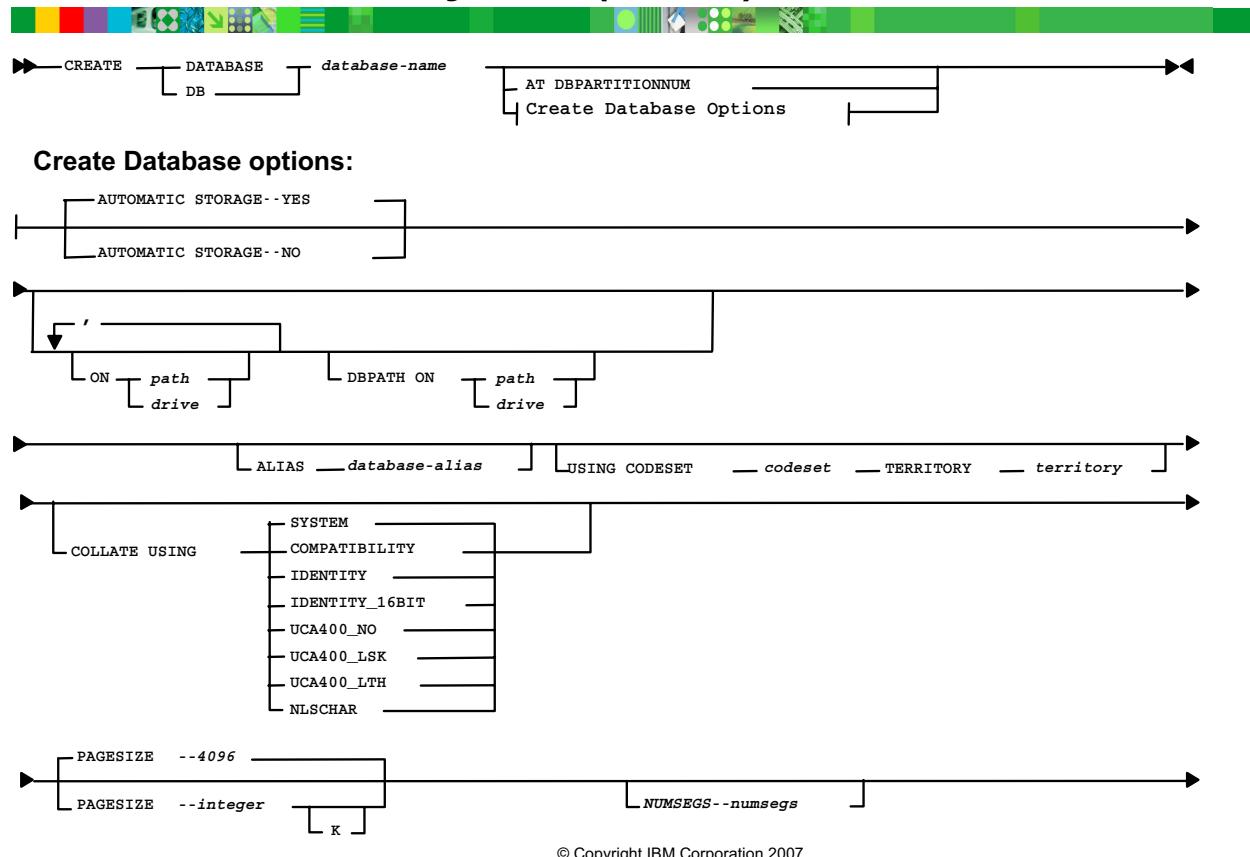


Figure 4-5. Create database syntax (1 of 3)

CF238.3

Notes:

The CREATE DATABASE command creates a database at the location defined by the ON parameter or at the default location. Only a user with SYSADM or SYSCTRL authority may create databases. A database can be created using CLP, or the Control Center, or through an application program invoking an API.

The CREATE DATABASE command initializes a new database, creates three initial table spaces, creates the system catalog tables, and allocates recovery logs.

Command Parameters:

database-name

Unique name to be assigned to the new database. 1-8 characters long.

- Must begin with alphabetic character, @, #, or \$
- May contain a-z, A-Z, 0-9, @, #, \$, or _ (underscore)

AT DBPARTITIONNUM

Specifies that the database is to be created only on the database partition that issues the command. You do not specify this option when you create a new database. You can use it to recreate a database partition that you dropped because it was damaged.

AUTOMATIC STORAGE

Specifies that automatic storage is being explicitly disabled or enabled for the database. The default value is YES. If the AUTOMATIC STORAGE clause is not specified, automatic storage is implicitly enabled by default.

- Must be NO or YES

path/drive

On UNIX platforms, the path on which to create the database can be specified. By default, the **DFTDBPATH** parameter specified in the database manager configuration file is used. On Windows platforms, the letter of the drive or path on which to create the database can be specified.

database-alias

An alias for the database. Defaults to the database name. The alias name is used by applications to connect to the database.

codeset

Specifies the code set to be used for data entered into this database.

territory

Specifies the territory to be used for data entered into this database.

COLLATE USING

Identifies the type of collating sequence to be used for this database.

SYSTEM indicates that the collating sequence is based on the current territory.

IDENTITY indicates that the collating sequence is the identity sequence, where strings are compared byte by byte.

dft-extentsize

Specifies the default extent size of table spaces in the database.

tblspace-defn

Specifies the definition of the table space which will be used to hold the catalog tables (CATALOG TABLESPACE — SYSCATSPACE), initial user table space (USER TABLESPACE — USERSPACE1), and initial temporary table space (TEMPORARY TABLESPACE — TEMPSPACE1).

“comment string”

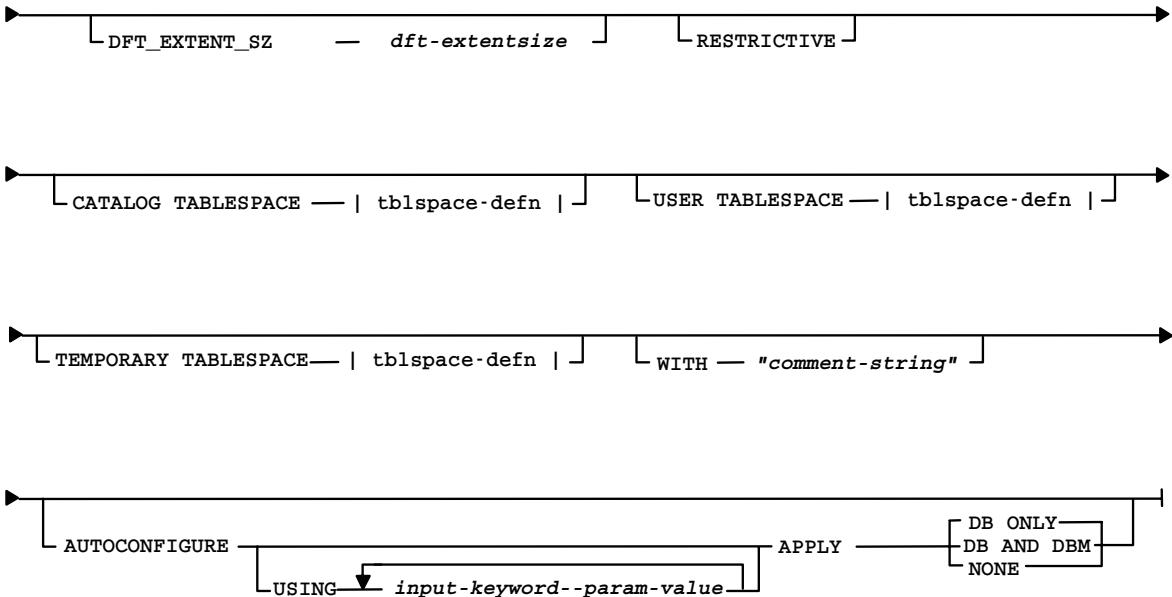
Up to 30-character comment describing the database.

PAGESIZE

Specifies the page size of the default buffer pool along with the initial table spaces (SYSCATSPACE, TEMPSPACE1, USERSPACE1) when the database is created. This also represents the default page size for all future CREATE BUFFERPOOL and CREATE TABLESPACE statements.

- Valid values for integer without the suffix K are 4096, 8192, 16384, or 32768.
- Valid values for integer with the suffix K are 4, 8, 16, or 32.
- At least one space is required between the integer and the suffix K. The default is a page size of 4096 bytes (4 K).

Create database syntax (2 of 3)

© Copyright IBM Corporation 2007

Figure 4-6. Create database syntax (2 of 3)

CF238.3

Notes:

More parts of the CREATE DATABASE statement are explained on this page.

DFT_EXTENT_SZ

Specifies the default extent size of table spaces in the database.

RESTRICTIVE

If the RESTRICTIVE option is present, it causes the RESTRICT_ACCESS database configuration parameter to be set to YES and no privileges are automatically granted to PUBLIC. If the RESTRICTIVE option is not present, then the RESTRICT_ACCESS database configuration parameter is set to NO and all of the following privileges are automatically granted to PUBLIC.

- CREATETAB
- BINDADD
- CONNECT
- IMPLSCHEMA
- EXECUTE with GRANT on all procedures in schema SQLJ

- EXECUTE with GRANT on all functions and procedures in schema SYSPROC
- BIND on all packages created in the NULLID schema
- EXECUTE on all packages created in the NULLID schema
- CREATEIN on schema SQLJ
- CREATEIN on schema NULLID
- USE on table space USERSPACE1
- SELECT access to the SYSIBM catalog tables
- SELECT access to the SYSCAT catalog views
- SELECT access to the SYSSTAT catalog views
- UPDATE access to the SYSSTAT catalog views

CATALOG TABLESPACE

Specifies the definition of the table space that will hold the catalog tables, SYSCATSPACE. If not specified and automatic storage is not enabled for the database, SYSCATSPACE is created as a System Managed Space (SMS) table space with *numsegs* number of directories as containers, and with an extent size of *dft_extentsize*.

If not specified and automatic storage is enabled for the database, SYSCATSPACE is created as an automatic storage table space with its containers created on the defined storage paths. The extent size of this table space is 4. Appropriate values for AUTORESIZE, INITIALSIZE, INCREASESIZE, and MAXSIZE are set automatically.

USER TABLESPACE

Specifies the definition of the initial user table space, USERSPACE1. If not specified and automatic storage is not enabled for the database, USERSPACE1 is created as an SMS table space with *numsegs* number of directories as containers and with an extent size of *dft_extentsize*.

If not specified and automatic storage is enabled for the database, USERSPACE1 is created as an automatic storage table space with its containers created on the defined storage paths. The extent size of this table space will be *dft_extentsize*. Appropriate values for AUTORESIZE, INITIALSIZE, INCREASESIZE, and MAXSIZE are set automatically.

TEMPORARY TABLESPACE

Specifies the definition of the initial system temporary table space, TEMPSPACE1. If not specified and automatic storage is not enabled for the database, TEMPSPACE1 is created as an SMS table space with *numsegs* number of directories as containers and with an extent size of *dft_extentsize*.

If not specified and automatic storage is enabled for the database, TEMPSPACE1 is created as an automatic storage table space with its containers created on the defined storage paths. The extent size of this table space is *dft_extentsize*.

WITH

Describes the database entry in the database directory. Any comment that helps to describe the database can be entered. Maximum length is 30 characters. A carriage

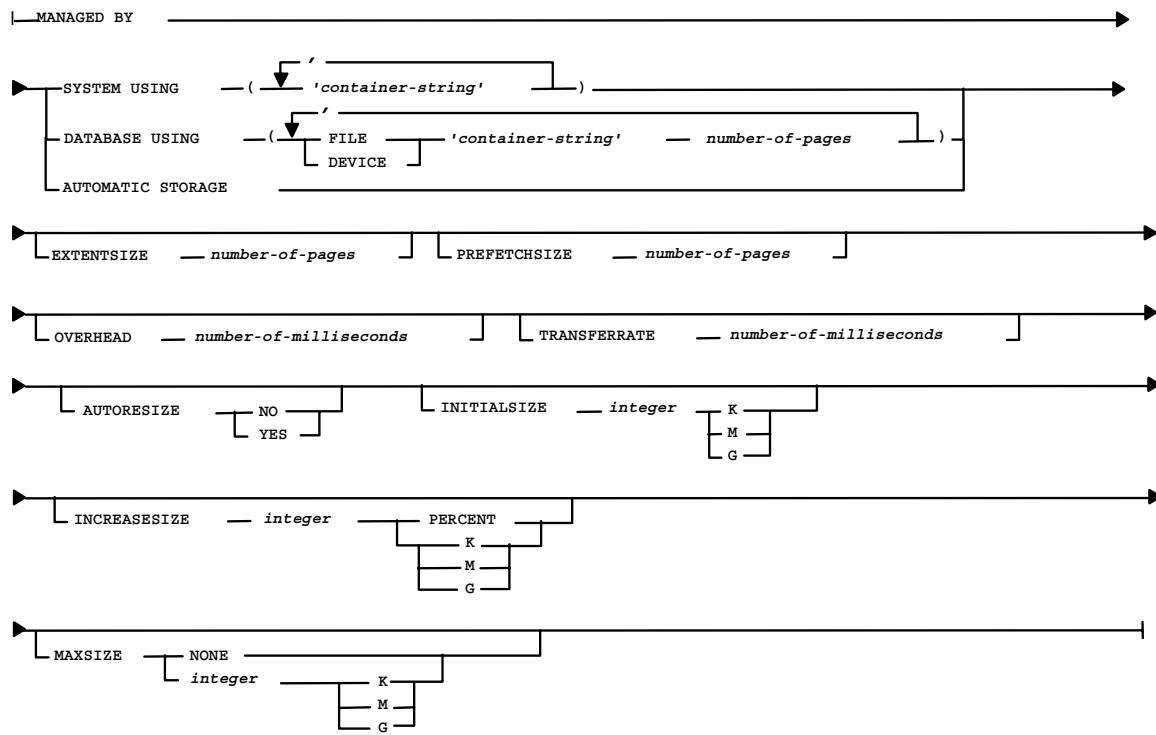
return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

AUTOCONFIGURE

Based on user input, calculates the recommended settings for buffer pool size, database configuration, and database manager configuration and optionally applies them. The Configuration Advisor is run by default when the CREATE DATABASE command is issued. The AUTOCONFIGURE option is needed only if you want to tweak the recommendations.

Create database syntax (3 of 3)

tblspace-defn:



© Copyright IBM Corporation 2007

Figure 4-7. Create database syntax (3 of 3)

CF238.3

Notes:

Part of the syntax of the CREATE DATABASE statement relation to the table space definitions is explained here, referencing the syntax diagram in the visual.

MANAGED BY

SYSTEM

Specifies that the table space is to be an SMS table space. When the type of table space is not specified, the default behavior is to create a regular table space.

DATABASE

Specifies that the table space is to be a DMS table space. When the type of table space is not specified, the default behavior is to create a large table space.

AUTOMATIC STORAGE

Specifies that the table space is to be an automatic storage table space. If automatic storage is not defined for the database, an error is returned (SQLSTATE 55060).

An automatic storage table space is created as either a system managed space (SMS) table space or a database managed space (DMS) table space. When DMS is chosen and the type of table space is not specified, the default behavior is to create a large table space. With an automatic storage table space, the database manager determines which containers are to be assigned to the table space, based upon the storage paths that are associated with the database.

EXTENTSIZE

Specifies the number of PAGESIZE pages that will be written to a container before skipping to the next container. The extent size value can also be specified as an integer value followed by K (for kilobytes) or M (for megabytes). If specified in this way, the floor of the number of bytes divided by the page size is used to determine the value for the extent size. The database manager cycles repeatedly through the containers as data is stored.

The default value is provided by the DFT_EXTENT_SZ database configuration parameter, which has a valid range of 2-256 pages.

PREFETCHSIZE

Specifies to read in data needed by a query prior to it being referenced by the query, so that the query need not wait for I/O to be performed.

The default value is provided by the DFT_PREFETCH_SZ database configuration parameter.

OVERHEAD

Specifies the I/O controller overhead and disk seek and latency time. This value is used to determine the cost of I/O during query optimization. The value of number-of-milliseconds is any numeric literal (integer, decimal, or floating point). If this value is not the same for all containers, the number should be the average for all containers that belong to the table space.

For a database that was created in Version 9 or later, the default I/O controller overhead and disk seek and latency time is 7.5 milliseconds. For a database that was migrated from a previous version of DB2 to Version 9 or later, the default is 12.67 milliseconds.

TRANSFERRATE

Specifies the time to read one page into memory. This value is used to determine the cost of I/O during query optimization. The value of number-of-milliseconds is any numeric literal (integer, decimal, or floating point). If this value is not the same for all containers, the number should be the average for all containers that belong to the table space.

For a database that was created in Version 9 or later, the default time to read one page into memory is 0.06 milliseconds. For a database that was migrated from a previous version of DB2 to Version 9 or later, the default is 0.18 milliseconds.

AUTORESIZE

Specifies whether or not the auto-resize capability of a DMS table space or an automatic storage table space is to be enabled. Auto-resizeable table spaces automatically increase in size when they become full. The default is NO for DMS table spaces and YES for automatic storage table spaces.

INITIALSIZE

Specifies the initial size, per database partition, of an automatic storage table space. This option is only valid for automatic storage table spaces. The integer value must be followed by K (for kilobytes), M (for megabytes), or G (for gigabytes). Note that the actual value used might be slightly smaller than what was specified, because the database manager strives to maintain a consistent size across containers in the table space. Moreover, if the table space is auto-resizeable and the initial size is not large enough to contain meta-data that must be added to the new table space, DB2 will continue to extend the table space by INCREASESIZE until there is enough space. If the table space is auto-resizeable, but the INITIALSIZE clause is not specified, the database manager determines an appropriate value.

INCREASESIZE

Specifies the amount, per database partition, by which a table space that is enabled for auto-resize will automatically be increased when the table space is full, and a request for space has been made. The integer value must be followed by:

- **PERCENT** to specify the amount as a percentage of the table space size at the time that a request for space is made. When PERCENT is specified, the integer value must be between 0 and 100 (SQLSTATE 42615).
- **K** (for kilobytes), **M** (for megabytes), or **G** (for gigabytes) to specify the amount in bytes.



Note

Note that the actual value used might be slightly smaller or larger than what was specified, because the database manager strives to maintain consistent growth across containers in the table space. If the table space is auto-resizeable, but the INCREASESIZE clause is not specified, the database manager determines an appropriate value.

MAXSIZE

Specifies the maximum size to which a table space that is enabled for auto-resize can automatically be increased. If the table space is auto-resizeable, but the MAXSIZE clause is not specified, the default is NONE.

Automatic storage — database

- Intended as a *single point of storage management* for table spaces
- Create a database and associate a set of storage paths with it
 - Create **AUTOMATIC STORAGE** table spaces
 - No explicit container definitions are provided
 - Containers automatically created across the storage paths
 - Growth of existing containers and addition of new ones managed by **DB2**
- Add storage paths to the database afterwards
- Redefine those storage paths during a database restore

© Copyright IBM Corporation 2007

Figure 4-8. Automatic storage — database

CF238.3

Notes:

By creating a database with AUTOMATIC STORAGE, the need for constant monitoring of space is eliminated. If maintenance needs to be performed on the storage spaces, storage paths can be added as needed.

Automatic storage syntax — database

Automatic storage databases

- Created as AUTOMATIC STORAGE YES by default (only on non-DPF servers)
- Disable automatic storage during database creation explicitly

Examples of automatic storage being disabled explicitly:

Examples of automatic storage being enabled either explicitly or implicitly:

```
CREATE DATABASE ASNODB1 AUTOMATIC STORAGE NO  
CREATE DATABASE ASNODB2 AUTOMATIC STORAGE NO ON X:
```

```
CREATE DATABASE DB1  
CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:  
CREATE DATABASE DB3 ON /data/path1, /data/path2  
CREATE DATABASE DB4 ON D:\StoragePath DBPATH ON C:
```

© Copyright IBM Corporation 2007

Figure 4-9. Automatic storage syntax — database

CF238.3

Notes:

By default, automatic storage is enable when creating a database. To disable this feature, set AUTOMATIC STORAGE to NO.

Automatic storage — create database examples



Examples:

```
CREATE DATABASE TESTDB1
```

- Automatic storage enabled: Yes
- Database and Storage path: *dftdbpath*

```
CREATE DATABASE TESTDB2
```

- ON /testdb2
- Automatic storage enabled: Yes
- Database and Storage path: */testdb2*

```
CREATE DATABASE TESTDB3
```

```
AUTOMATIC STORAGE YES
```

- Automatic storage enabled: Yes
- Database path: *dftdbpath*
- Storage path: *dftdbpath*

```
CREATE DATABASE TESTDB4
```

```
AUTOMATIC STORAGE YES ON /dbdir
```

- Automatic storage enabled: Yes
- Database path: */dbdir*
- Storage path: */dbdir*

```
CREATE DATABASE TESTDB5
```

```
ON /db2/dir1,/db2/dir2,/db2/dir3
```

- Automatic storage enabled: Yes
- Database path: */db2/dir1*
- Storage paths: */db2/dir1, /db2/dir2, /db2/dir3*

```
CREATE DATABASE TESTDB6
```

```
ON D: \AS_PATH DBPATH ON C:
```

- Automatic storage enabled: Yes
- Database path: *C:*
- Storage path: *D:\AS_PATH*

© Copyright IBM Corporation 2007

Figure 4-10. Automatic storage — create database examples

CF238.3

Notes:

Examples of creating databases with and without automatic storage are shown above:

- **TESTDB1** — Set to automatic storage, and uses a default database path
- **TESTDB2** — Set to automatic storage, but uses a specified database path
- **TESTDB3** — Set to automatic storage, using the default database path and a default storage path
- **TESTDB4** — Set to automatic storage, using the same specified database path and storage path
- **TESTDB5** — Set to automatic storage, using the specified database path and specified storage paths
- **TESTDB6** — Set to automatic storage, using the specified database path and a different specified storage path

Automatic Storage provisioning — syntax

```

1. CREATE DATABASE DB1
   AUTOMATIC STORAGE YES

2. CREATE DATABASE DB3
   AUTOMATIC STORAGE YES
   ON 'C:\', 'C:\DB2\data\path1',
   'C:\DB2\data\path2'
   DBPATH ON 'C:\'

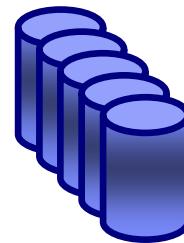
3. CREATE TABLESPACE TS2
   MANAGED BY AUTOMATIC STORAGE

4. CREATE TEMPORARY TABLESPACE TEMPTS

5. CREATE USER TEMPORARY TABLESPACE USRTMP
   MANAGED BY AUTOMATIC STORAGE

6. CREATE TABLESPACE TS1
   INITIALSIZE 500 K
   INCREASESIZE 100 K
   MAXSIZE 100 M

```



© Copyright IBM Corporation 2007

Figure 4-11. Automatic Storage provisioning — syntax

CF238.3

Notes:

In the information shown above, the first statement is the simplest, declaring AUTOMATIC STORAGE for the whole database.

The following statements provide a variety of syntax for both databases and table spaces, and paths created are shown in the table.

Table 1: Example storage paths

	Table space	Path	Type
1	SYSCATSPACE	C:\DB2\NODE0000\DB1\T0000000	Regular
	TEMPSPACE1	C:\DB2\NODE0000\DB1\T0000001	System temporary
	USERSPACE1	C:\DB2\NODE0000\DB1\T0000002	Large
	SYSTOOLSPACE	C:\DB2\NODE0000\DB1\T0000003	Large

Table 1: Example storage paths

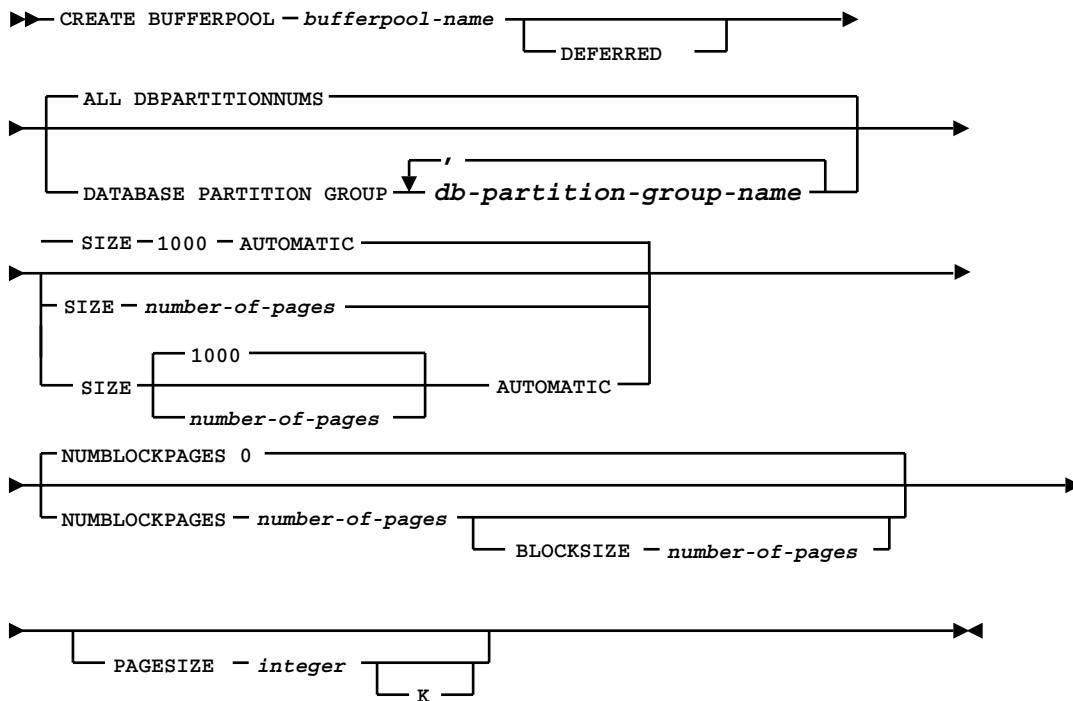
2	SYSCATSPACE	C:\DB2\NODE000\DB3\T0000000 C:\DB2\data\path1\DB2\NODE000\DB3\T0000000 C:\DB2\data\path2\DB2\NODE000\DB3\T0000000	Regular
	TEMPSPACE1	C:\DB2\NODE000\DB3\T0000001 C:\DB2\data\path1\DB2\NODE000\DB3\T0000001 C:\DB2\data\path2\DB2\NODE000\DB3\T0000001	System temporary
	USERSPACE1	C:\DB2\NODE000\DB3\T0000002 C:\DB2\data\path1\DB2\NODE000\DB3\T0000002 C:\DB2\data\path2\DB2\NODE000\DB3\T0000002	Large
3	TS2	C:\DB2\NODE000\SAMPLE\T0000006	Large
4	TEMPTS	C:\DB2\NODE000\SAMPLE\T0000008	System temporary
5	UDRTMP	C:\DB2\NODE000\SAMPLE\T0000009	User temporary
6	TS1	C:\DB2\NODE000\SAMPLE\T0000010	Large

Note that in the second example, several locations are used for storage:

C:\DB2\NODE000\DB3, C:\DB2\data\path1\DB2\NODE000\DB3, and C:\DB2\data\path2\DB2\NODE000\DB3.

Create bufferpool syntax

- Requires SYSCTRL or SYSADM authority



© Copyright IBM Corporation 2007

Figure 4-12. Create bufferpool syntax

CF238.3

Notes:

The CREATE BUFFERPOOL statement creates a new buffer pool to be used by the database manager.

Once a buffer pool has been created, it can be assigned to a table space using the ALTER TABLESPACE command. When creating a table space, a buffer pool can be assigned to the table space.

The authorization ID of the statement must have SYSCTRL or SYSADM authority.

Default table spaces and buffer pool



- Default table spaces

- SYSCATSPACE

- Stores system catalog tables and views

- TEMPSPACE1

- Used by the system to store temporary activity, like sorts

- USERSPACE1

- First table space for users to create data and index structures

- Default buffer pool

- IBMDEFAULTTBP

- Created when the database is created, and used by data from all default table spaces

© Copyright IBM Corporation 2007

Figure 4-13. Default table spaces and buffer pool

CF238.3

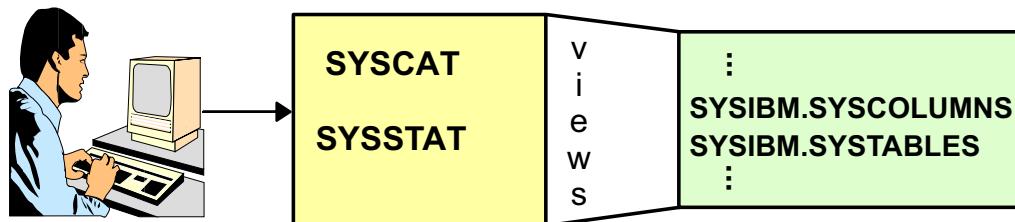
Notes:

When the database is initially created, a set of three table spaces will be defined:

- SYSCATSPACE for the system catalog tables
- USERSPACE1 for user-defined tables
- TEMPSPACE1 for temporary work area

If you do not specify any table space parameters on the CREATE DATABASE command, the database manager will create DMS table spaces for SYSCATSPACE and USERSPACE1, and SMS table space for TEMPSPACE1. The default extent size for SYSCATSPACE is 4 pages, and 32 pages for TEMPSPACE1 and USERSPACE1.

System Catalog tables and views



© Copyright IBM Corporation 2007

Figure 4-14. System Catalog tables and views

CF238.3

Notes:

A set of system catalog tables is created and maintained for each database. These tables contain information about the definitions of the database objects and also security information about the type of access users have to these objects.

The database manager creates and maintains two sets of catalog views. All of the system catalog views are created when a database is created with the CREATE DATABASE command. The catalog views cannot be explicitly created or dropped. The views are within the SYSCAT schema and select privilege on all views is granted to PUBLIC by default. A second set of views formed from a subset of those within the SYSCAT schema contains statistical information used by the SQL optimizer. The views within the SYSSTAT schema contain some updatable columns.

Table 3-1. Catalog Views as documented in *SQL Reference*

Catalog View	Description
SYSCAT.ATTRIBUTES	attributes of structured data types
SYSCAT.BUFFERPOOLDBPARTITIONS	buffer pool size on database partition
SYSCAT.BUFFERPOOLS	buffer pool configuration on database partition group
SYSCAT.CASTFUNCTIONS	cast functions
SYSCAT.CHECKS	check constraints
SYSCAT.COLAUTH	column privileges
SYSCAT.COLCHECKS	columns referenced by check constraints
SYSCAT.COLDIST	detailed column options
SYSCAT.COLGROUPCOLS	detailed column group columns
SYSCAT.COLGROUPS	detailed column group statistics
SYSCAT.COLIDENTATTRIBUTES	attributes of identity column
SYSCAT.COLOPTIONS	detailed column options
SYSCAT.COLUMNS	columns
SYSCAT.COLUSE	columns used in dimensions
SYSCAT.CONSTDEP	constraint dependencies
SYSCAT.DATATYPES	data types
SYSCAT.DBAUTH	authorities on database
SYSCAT.DBPARTITIONGROUPDEF	database partition group database partitions
SYSCAT.DBPARTITIONGROUPS	database partition group definitions
SYSCAT.EVENTMONITORS	event monitor definitions
SYSCAT.EVENTS	events currently monitored
SYSCAT.EVENTTABLES	events currently monitored
SYSCAT.FULLHIERARCHIES	hierarchies (types, tables, views)
SYSCAT.FUNCMAPOPTIONS	function mapping options
SYSCAT.FUNCMAPPARMOPTIONS	function mapping parameter options
SYSCAT.FUNCMAPPINGS	function mapping
SYSCAT.HIERARCHIES	hierarchies (types, tables, views)
SYSCAT.INDEXAUTH	index privileges
SYSCAT.INDEXCOLUSE	index columns
SYSCAT.INDEXDEP	index dependencies
SYSCAT.INDEXES	indexes
SYSCAT.INDEXEXPLOITRULES	index exploitation
SYSCAT.INDEXEXTENSIONDEP	index extension dependencies

Table 3-1. Catalog Views as documented in SQL Reference

SYSCAT.INDEXEXTENSIONMETHODS	index extension search methods
SYSCAT.INDEXEXTENSIONPARMS	index extension parameters
SYSCAT.INDEXEXTENSIONS	index extensions
SYSCAT.INDEXOPTIONS	index options
SYSCAT.KEYCOLUSE	columns used in keys
SYSCAT.NAMEMAPPINGS	object mapping
SYSCAT.PACKAGEAUTH	package privileges
SYSCAT.PACKAGEDEP	package dependencies
SYSCAT.PACKAGES	packages
SYSCAT.PARTITIONMAPS	partitioning maps
SYSCAT.PASSTHRUAUTH	pass-through privileges
SYSCAT.PREDICATESPEC	predicate specifications
SYSCAT.PROCOPTIONS	procedure options
SYSCAT.PROCPARMOPTIONS	procedure parameter options
SYSCAT.REFERENCES	referential constraints
SYSCAT.REVTYPEMAPPINGS	reverse data type mapping
SYSCAT.ROUTINEAUTH	routine privileges
SYSCAT.ROUTINEDEP	function, method, routine dependencies
SYSCAT.ROUTINEPARAMS	function, method, procedure, routine parameters
SYSCAT.ROUTINES	functions, methods, procedures, routines, stored procedures, user-defined functions
SYSCAT.SCHEMAUTH	schema privileges
SYSCAT.SCHEMATA	schemas
SYSCAT.SEQUENCEAUTH	sequence privileges
SYSCAT.SEQUENCES	sequences
SYSCAT.SERVEROPTIONS	server options
SYSCAT.SERVERS	system servers
SYSCAT.STATEMENTS	statements in packages
SYSCAT.TABAUTH	table privileges
SYSCAT.TABCONST	table constraints
SYSCAT.TABDEP	table dependencies
SYSCAT.TABLES	tables
SYSCAT.TABLESPACES	table spaces
SYSCAT.TABOPTIONS	remote table options
SYSCAT.TBSPACEAUTH	table spaces use privileges
SYSCAT.TRANSFORMS	transforms

Table 3-1. Catalog Views as documented in *SQL Reference*

SYSCAT.TRIGDEP	trigger dependencies
SYSCAT.TRIGGERS	triggers
SYSCAT.TYPENAMEPINGS	type mapping
SYSCAT.USEROPTIONS	server options values
SYSCAT.VIEWS	views
SYSCAT.WRAPOPTIONS	wrapper options
SYSCAT.WRAPPERS	wrappers
SYSIBM.SYSDUMMY1	provides DB2 UDB for OS/390 compatibility

Table 3-2. Updatable Catalog Views as documented in *SQL Reference*

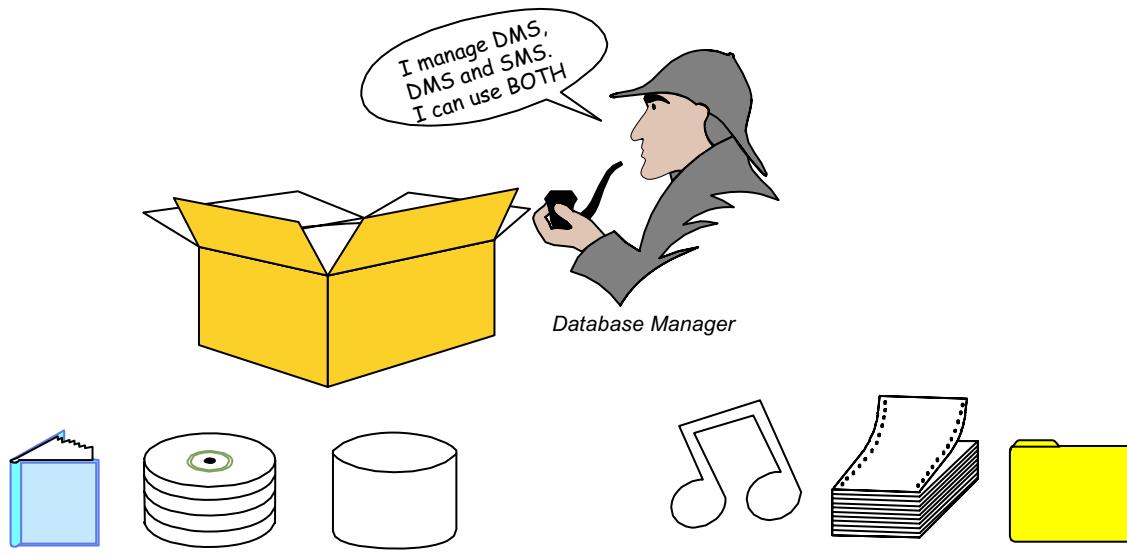
Catalog View	Description
SYSSTAT.COLDIST	detailed column statistics
SYSSTAT.COLGROUPS	detailed column group statistics
SYSSTAT.COLUMNS	columns
SYSSTAT.INDEXES	indexes
SYSSTAT.ROUTINES	routines
SYSSTAT.TABLES	tables

4.2 Table spaces and containers

DMS versus SMS table spaces



- SMS - Database Manager manages data using the operating system
- DMS - Managed by Database Manager directly



© Copyright IBM Corporation 2007

Figure 4-15. DMS versus SMS table spaces

CF238.3

Notes:

Table spaces are a logical level between the database and the tables stored in that database. There are two types of table spaces that can be created, SMS and DMS. A database can have any combination of SMS and DMS table spaces.

In an **SMS (System Managed Space)** table space, the operating system's file system manager allocates and manages the space where the table is stored. The storage model typically consists of many files, representing table objects, stored in the file system space. The user decides on the location of the files; DB2 controls their names; and the file system is responsible for managing them. By controlling the amount of data written to each file, the database manager distributes the data evenly across the table space containers. By default, the initial table spaces created at database creation time are SMS.

In a **DMS (Database Managed Space)** table space, the database manager controls the storage space. The storage model consists of a limited number of devices or files whose space is managed by DB2. The database administrator decides which devices and files to use, and DB2 manages the space on those devices and files. The table space is essentially an implementation of a special-purpose file system designed to best meet the needs of the database manager.

In DB2 9, auto growth and Automatic Storage are enabled as default.

Auto growth

A table space will auto-extend when it is full and more space is needed. However, only those containers that are part of the last range in the table space map will grow. This ensures that a rebalance will never take place as part of an autoresize.

- Autoresize is a characteristic of DMS tablespaces, which also applies to the REGULAR and LARGE tablespaces using AUTOMATIC STORAGE.

Auto-growth will stop when any of the following happen:

- The value specified for MAXSIZE is reached -or-
- DB2's table space size limit has been reached (if MAXSIZE is NONE) -or-
- One of the containers in the last range cannot grow any further
 - DB2 will not extend the others in the last range

To continue growth, you can add space to the file system, or add a new stripe set.

Automatic Storage

In DB2 9, automatic storage support is added for multi-partition databases. In addition, the Control Center has been enhanced to let you create databases that use automatic storage, and to enable you to add storage paths to existing databases.

By creating a database with AUTOMATIC STORAGE, the need for constant monitoring of space is eliminated. If maintenance needs to be performed on the storage spaces, storage paths can be added as needed.

Default table spaces

The type of table space for the three default table spaces has changed with DB2 9.

The Syscatspace and Userspace1 table spaces are created as DMS table spaces by default, while the Tempspace1 table space is created as SMS.

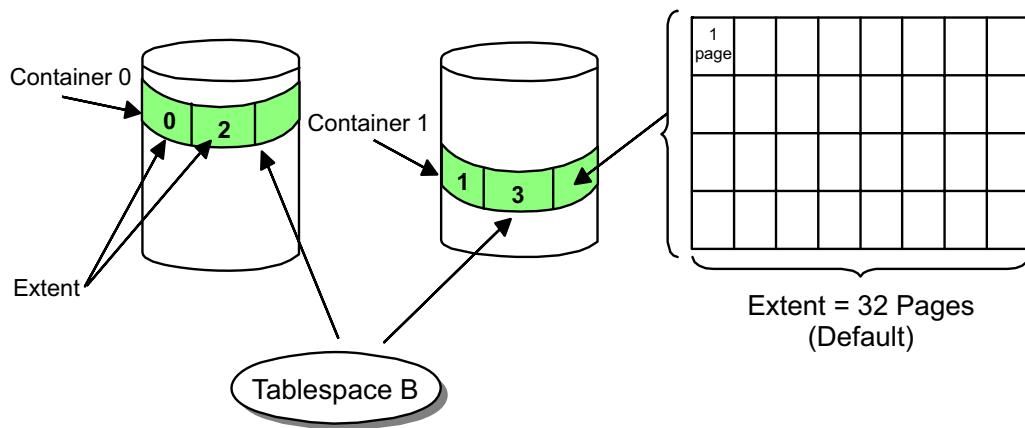
The default page size for these table spaces is 4 K.

Default storage paths (Windows):

- Syscatspace — C:\DB2\NODE0000\SAMPLE\T0000000 (DMS-file)
- Userspace1 — C:\DB2\NODE0000\SAMPLE\T0000002 (DMS-file)
- Tempspace1 — C:\DB2\NODE0000\SAMPLE\T0000001 (SMS-folder)

Writing to containers

- DFT_EXTENT_SZ Defined at database level
- EXTENTSIZE Defined at table space level
- Data written in round-robin manner



© Copyright IBM Corporation 2007

Figure 4-16. Writing to containers

CF238.3

Notes:

An extent is an allocation of contiguous space within a container of a table space. Extents are allocated to a single database object and consist of multiple pages. The default extent size is 32 pages, but a different value can be specified when creating a table space.

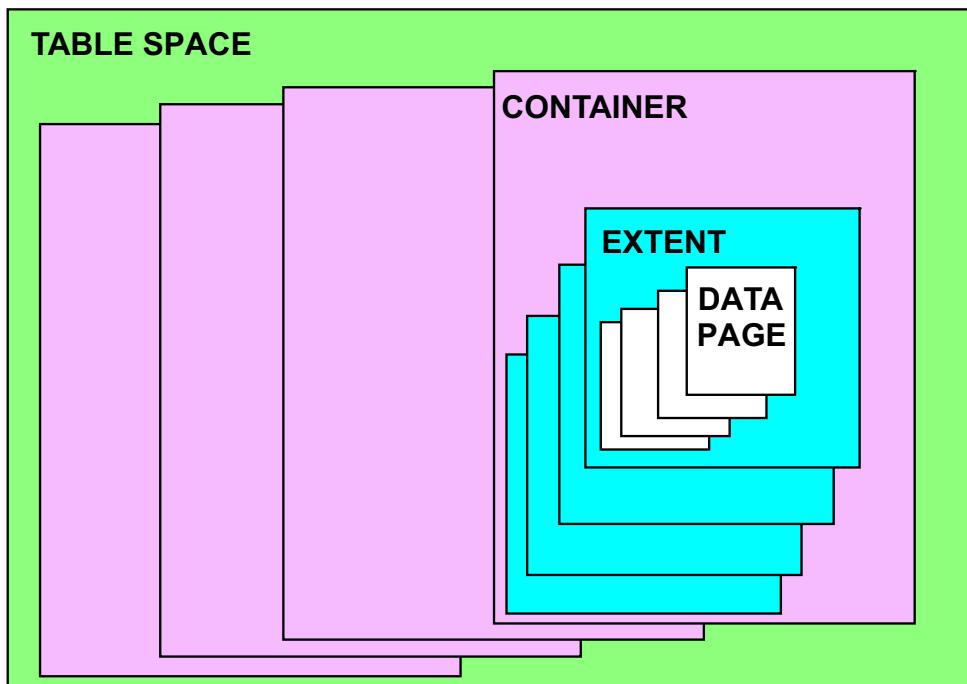
Data for an object is stored on a container by extents. When data for an object is written, the database system stripes the data across all the containers in the table space based on the extent size.

DFT_EXTENT_SZ is a database configuration parameter. If you do not specify an extent size when the table space is created, DFT_EXTENT_SZ will be used. The default size for DFT_EXTENT_SZ is 32 pages. If you do not alter this value or explicitly indicate an extent size when you create a table space, all your table spaces within the database will have this default value. The range of values for DFT_EXTENT_SZ is between two and 256 pages.

You can specify extent size at the table space level. This can be done at table space creation with the parameter EXTENTSIZE. Carefully determine the correct size, as once it is set for a table space, it cannot be altered. This size can have an impact on space utilization and performance.

The database manager will try to evenly distribute the table among containers. In doing so, the database manager writes an extent of pages to each container before writing to the next container. Once the database manager has written an extent to all the containers allocated to a table space, it will write the next extent to the first container written to in that table space. This round-robin process of writing to the containers is designed to balance the workload across the containers of the table space.

Table space, container, extent, page



© Copyright IBM Corporation 2007

Figure 4-17. Table space, container, extent, page

CF238.3

Notes:

A table space is made up of containers.

A container is made up of extents.

An extent is made up of data pages.

DMS table space — minimum requirements



Table Space Creation	container tag	1 extent per container
	header for table space	1 extent
	space map	1 extent
	object table data	1 extent
		4 extents
Table (Object) Creation	extent map	1 extent
	extent for data	1 extent
	Minimum space required in containers = (for first object)	
	If EXTENTSIZE = 32 pages, then $6 \times 32 = 192$ pages needed to be provided in containers	

© Copyright IBM Corporation 2007

Figure 4-18. DMS table space — minimum requirements

CF238.3

Notes:

All indexes for a table are considered to be a single object.

A certain amount of overhead is required in a DMS table space. The space required is:

- One extent per container for a container tag
- Three extents per table space for the table space meta data
- Two initial extents per object in the table space: one for the extent map for the object, one for the data for the object.

When a DMS table space is created, there must be enough space in the containers for six extents of the table space, otherwise the table space will not be created.

Note: Once DB2 removes the one extent per container for the container tag, it will make the number of usable pages a multiple of the extent size. So, for optimal space utilization, allocate an even multiple of the extent size in pages in each container so no space is wasted.

Defining table spaces



- When defining a table space, you can
 - Specify how the table space will be used
 - Specify the management of the table space
 - Specify containers and their description
 - Specify extent size
 - Specify performance characteristics

© Copyright IBM Corporation 2007

Figure 4-19. Defining table spaces

CF238.3

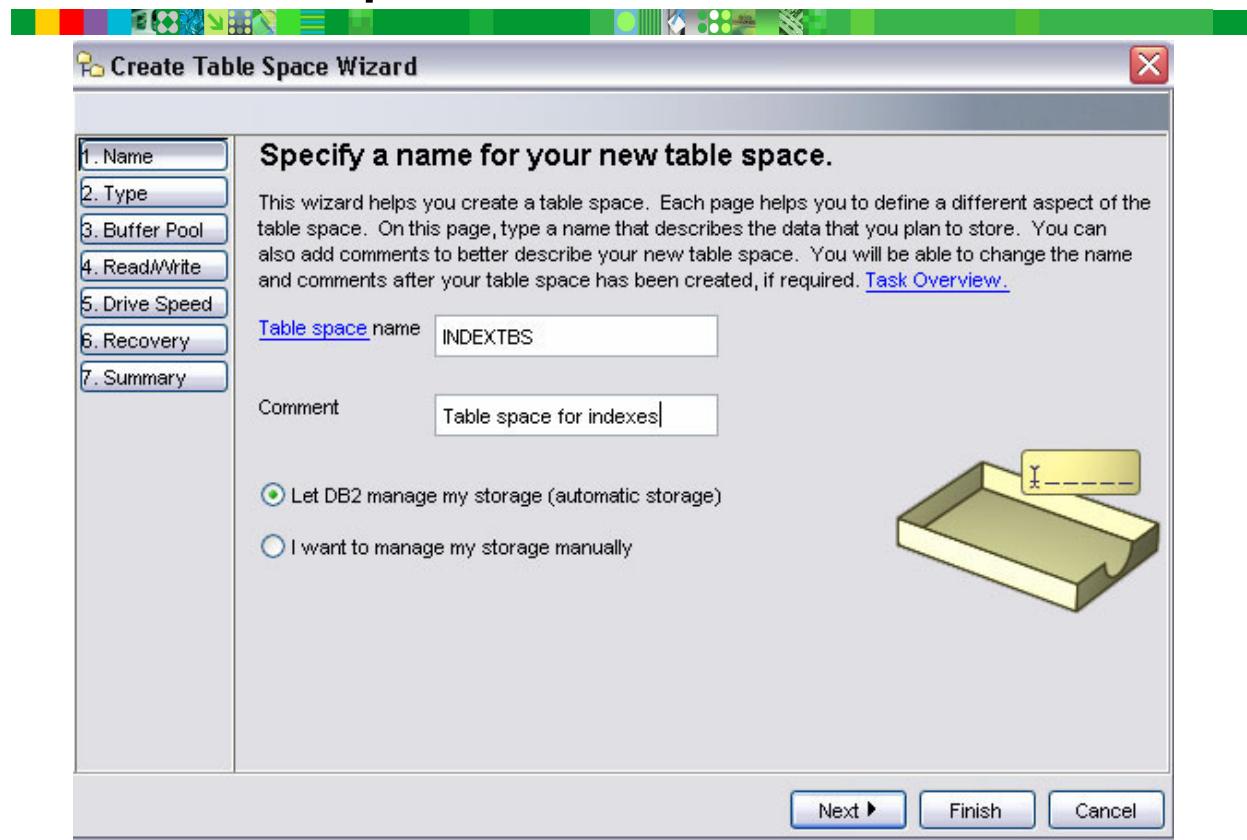
Notes:

When defining a table space, the following information can be supplied:

- How the table space will be used: REGULAR, LARGE (DMS only), or TEMPORARY
- The management of the table space: SMS or DMS
- Whether or not to use AUTOMATIC STORAGE
- The type, location, and size (DMS) of the containers
- The extent size
- Performance characteristics, such as buffer pool assignment, overhead, transfer rate, and prefetch size

The next pages will show how you can create table spaces within a database. We will show both SMS and DMS table spaces being created in the *ourdb* database. The examples will illustrate these different ways of creating table spaces. Some values will be allowed to default.

Create Table Space Wizard



© Copyright IBM Corporation 2007

Figure 4-20. Create Table Space Wizard

CF238.3

Notes:

The Create Table Space Wizard makes creating a table space easy. There are a set of wizard pages that you fill out:

Name

On this page, you provide a name for your new table space. You can also provide a comment to describe the table space.

Type

Use this page to select the type of table space you want to create. You can use a regular table space for any type of data except temporary tables. However, you will want a large table space if you are going to store long or LOB data such as images, audio, video, or long text fields. You can also choose to create a table space that can be used by the database manager for temporary tables created during sorts, joins, and other operations, or to create a table space that will be used to store user-defined temporary tables.

Database Partition Group

This page is only available if you are creating a table space on a partitioned database system. Use this page to select the database partition group that includes the database partitions on which the table space will reside. You can also launch the Create Database Partition Group window if you need to create a new one.

Buffer Pool

Use this page to specify the name of the buffer pool to be used for tables in this table space. You can choose from a list of existing buffer pools or you can launch the Create Buffer Pool window to create a new one. The page size of the buffer pool must match the page size specified for the table space. The database partition group of the table space must be defined for the buffer pool.

Space Management

On this page you select how your table space should be managed. In a System Managed Space (SMS) table space, the operating system's file system manager allocates and manages the space where the table is to be stored. The storage model typically consists of many files, representing table objects, stored in the file system space. In a Database Managed Space (DMS) table space, the database manager controls the storage space. The storage model consists of a limited number of devices or files, whose space is managed by DB2.

Containers

Use this page to identify one or more containers that will belong to the table space and into which the table space's data will be stored. All containers must be unique across all databases; a container can belong to only one table space. The size of the containers can differ; however, optimal performance is achieved when all containers are the same size.

Read/Write

Based on the information you provide about the average size of a table to be stored in the table space, the wizard will recommend the prefetch size and extent size settings for your table space. You can modify these settings. Note that you can modify the prefetch size after the table space has been created, but you cannot modify the extent size.

Drive Speed

Use this page to specify the type of system you are working on (for example, server, RAID environment, or storage area network). Based on the information you provide, the wizard will estimate your hard drive specifications, such as the average seek time, rotation speed, and transfer rate. If you know the exact specifications, you can enter them instead of using the wizard's estimates. Based on the specifications, the wizard will calculate values for overhead and transfer.

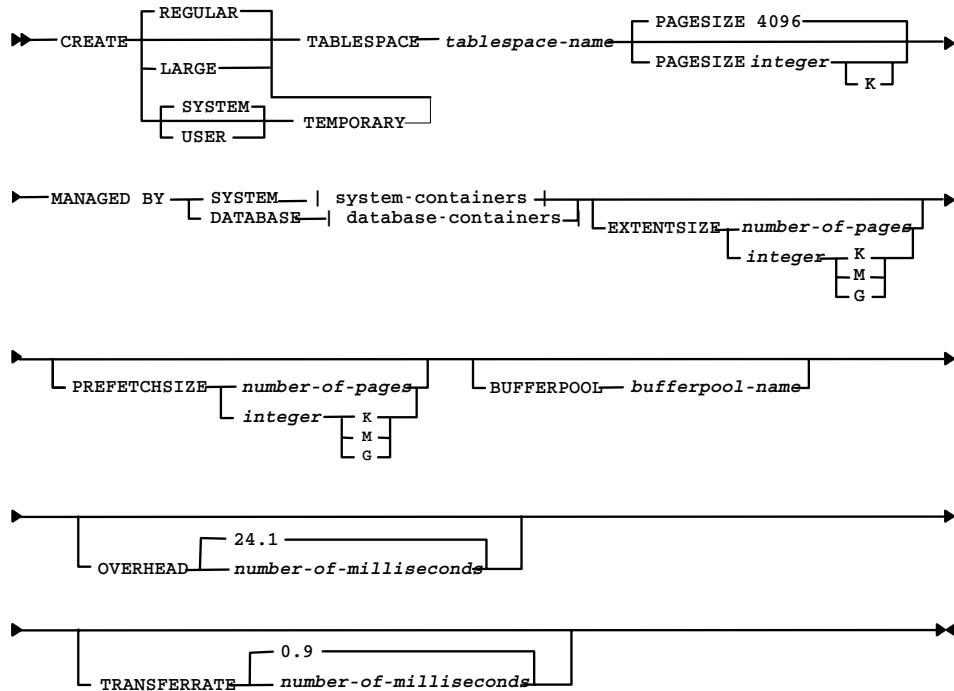
Recovery

This page allows you to enable dropped table recovery. When this option is enabled, you can recover a dropped table by restoring your table space and rolling forward. If you do not enable this option, you can still recover a dropped table by restoring the entire database and rolling forward.

Summary

This page allows you to review the decisions that you made for your table space, and to view the command that will create the table space.

CREATE TABLESPACE syntax (1 of 2)



© Copyright IBM Corporation 2007

Figure 4-21. CREATE TABLESPACE syntax (1 of 2)

CF238.3

Notes:

The CREATE TABLESPACE syntax diagram starts on this visual and ends on the next visual.

The CREATE TABLESPACE statement creates a new table space within the database, assigns containers to the table space, and records the table space definition and attributes in the catalog. Only users with SYSADM or SYSCtrl authority can create table spaces.

REGULAR

Stores all data except for temporary tables.

LARGE

Stores long or LOB table columns, or indexes. Can only be defined as a DMS table space.

In DB2 9, with large RID support, regular data can be stored in Large table spaces.

SYSTEM TEMPORARY

Stores temporary tables (work areas used by the database manager to perform operations such as sorts or joins). The keyword SYSTEM is optional. A database must always have at least one SYSTEM TEMPORARY table space, as temporary tables can only be stored in such a table space. A SYSTEM TEMPORARY table space is created when a database is created. A database can have more than one temporary table space. Temporary objects are allocated between the temporary table spaces in a round-robin fashion.

Recommendation: Create as few temporary table spaces as possible. It is better to put all of the containers into a single table space rather than dividing them up among multiple table spaces. Any given activity that uses a temporary table space (for example, a sort) will use only one temporary table space. If the temporary table space it uses is too small, the activity will fail.

USER TEMPORARY

Stores declared global temporary tables. No USER TEMPORARY table spaces exist when a database is created. At least one user temporary table space should be created with appropriate USE privileges to allow definition of declared temporary tables.

tablespace-name

Names the table space. This is a one-part name. It can be up to 18 characters in length.

PAGESIZE

Defines the size of pages used for the table space. Valid values are 4 KB, 8 KB, 16 KB, and 32 KB, or the equivalent integer values.

EXTENTSIZE number-of-pages

Specifies the number of PAGESIZE pages that will be written to a container before skipping to the next container. The database manager cycles repeatedly through the containers as data is stored.

The default value is provided by the DFT_EXTENT_SZ configuration parameter.

PREFETCHSIZE number-of-pages

Specifies the number of PAGESIZE pages that will be read from the table space when data prefetching is being performed. Prefetching reads in data needed by a query prior to it being referenced by the query, so the query need not wait for I/O to be performed.

The default value is provided by the DFT_PREFETCH_SZ configuration parameter.

BUFFERPOOL bufferpool-name

The name of the buffer pool used for tables in this table space. The buffer pool must exist. If not specified, the default buffer pool, IBMDEFAULTBP, is used.

OVERHEAD number-of-milliseconds

Any numeric literal that specifies the I/O controller overhead and disk seek and latency time, in milliseconds. The number should be an average for all containers that belong to the table space, if not the same for all containers.

TRANSFERRATE number-of-milliseconds

Any numeric literal that specifies the time to read one page into memory, in milliseconds. The number should be an average for all containers that belong to the table space, if not the same for all containers.

CREATE TABLESPACE syntax (2 of 2)



system-containers:

```
|---USING---( 'container-string' )---
```

database-containers:

© Copyright IBM Corporation 2007

Figure 4-22. CREATE TABLESPACE syntax (2 of 2)

CF238.3

Notes:

MANAGED BY SYSTEM

Specifies that the table space is to be a SMS table space.

USING ('container-string')

Identifies one or more containers that will belong to the table space and into which the table space's data will be stored. Each container string can be an absolute or relative directory name.

MANAGED BY DATABASE

Specifies that the table space is to be a DMS table space.

USING (FILE|DEVICE 'container-string' number-of-pages)

Identifies one or more containers that will belong to the table space and into which the table space's data will be stored. The type of the container (either FILE or DEVICE) and its size (in PAGESIZE pages) are specified.

For a FILE container, the string must be an absolute or relative file name.

For a DEVICE container, the string must be a device name. The device must already exist. Note that device containers are not supported on Windows 98, or on Linux on OS/390.

All containers must be unique across all databases; a container can belong to only one table space. The size of the containers can differ; however, optimal performance is achieved when all containers are the same size.

Automatic storage — table space examples

- Syntax for CREATE and ALTER TABLESPACE:

```
CREATE TABLESPACE <tsName> [MANAGED BY AUTOMATIC STORAGE]
[INITIALSIZE integer {K|M|G}]
[AUTORESIZE {NO|YES}] [INCREASESIZE integer {PERCENT|K|M|G}]
[MAXSIZE {NONE | integer {K|M|G}}]
```

- Default initial size is 32 MB
- Default max size is none
- Default increase size is determined by DB2, which might change over the life of the table space
- Examples:

```
CREATE TABLESPACE USER1
CREATE TEMPORARY TABLESPACE TEMPTS
CREATE TABLESPACE MYTS INITIALSIZE 100 M MAXSIZE 1 G
CREATE LARGE TABLESPACE LRGTS INITIALSIZE 5 G AUTORESIZE NO
CREATE REGULAR TABLESPACE USER2 INITIALSIZE 500 M
```

© Copyright IBM Corporation 2007

Figure 4-23. Automatic storage — table space examples

CF238.3

Notes:

When creating a table space in a database that is not enabled for automatic storage, the MANAGED BY SYSTEM or MANAGED BY DATABASE clause must be specified. Using these clauses results in the creation of a system managed space (SMS) table space or database managed space (DMS) table space respectively. An explicit list of containers must be provided in both cases.

If a database is enabled for automatic storage, another choice exists. The MANAGED BY AUTOMATIC STORAGE clause might be specified, or the MANAGED BY clause might be left out completely (which implies automatic storage). No container definitions are provided in this case because the DB2 database manager assigns the containers automatically.

In the CREATE TABLESPACE statements shown above:

USER1 — Created with automatic storage, with an initial size of 32 MB and will auto-resize.

TEMPTS — Created with automatic storage, as a SMS managed temporary tablespace with one directory on each automatic storage path. SMS managed tablespaces do not have a MAXSIZE limit.

MYTS — Created with automatic storage, with an initial size of 100 MB and will auto-resize until it reaches 1 GB.

LRGTS — Created with automatic storage, with an initial size of 5 GB and will not auto-resize.

USER2 — Created the automatic storage with an initial size of 500 MB and will auto-resize.

4.3 Gathering system information

Get snapshot for tablespaces command

- Use GET SNAPSHOT FOR TABLESPACES ON
 - LIST TABLESPACES does not provide all of the information available
- Example of GET SNAPSHOT FOR TABLESPACES ON SAMPLE

```
<database> command:
Tablespace name = SYSCATSPACE
Tablespace ID = 0
...
Automatic Prefetch size enabled = Yes
Buffer pool ID currently in use = 1
Buffer pool ID next startup = 1
Using automatic storage = Yes
Auto-resize enabled = Yes
...
Increase size (bytes) = AUTOMATIC
```

- Note that Using automatic storage = Yes and Increase size (bytes) = AUTOMATIC are shown in the snapshot.

© Copyright IBM Corporation 2007

Figure 4-24. Get snapshot for tablespaces command

CF238.3

Notes:

To identify which table spaces are enabled for automatic storage, use snapshots such as GET SNAPSHOT FOR TABLESPACES ON. Do not use the LIST TABLESPACES command as it identifies a table space as SMS or DMS, but does not identify which table spaces are enabled for automatic storage.

Snapshot monitors may be set up by:

- Set monitor switches on/off:

UPDATE DBM CFG USING DFT_MON_BUFPOOL ON or	(instance level)
---	------------------

UPDATE MONITOR SWITCHES USING BEFFERPOOL ON	(application level)
---	---------------------

- Determine how the monitor switches are set:

db2 get dbm monitor switches or	(instance level)
------------------------------------	------------------

db2 get monitor switches	(application level)
--------------------------	---------------------

- **To gather tablespace information:**

get snapshot for tablespaces on <database_name>

- **Reset monitor switches:**

reset monitor for database <database_name>

or

reset monitor all

Example of LIST TABLSPACES command:

Tablespace ID	= 0
Name	= SYSCATSPACE
Type	= Database managed space
Contents	= All permanent data. Regular table space.
State	= 0x0000

Detailed explanation:

Normal

Note that no information about automatic storage is specified.

Example of GET SNAPSHOT FOR TABLESPACES ON <database> command:

Tablespace name	= SYSCATSPACE
Tablespace ID	= 0
Tablespace Type	= Database managed space
Tablespace Content Type	= All permanent data. Regular table space.
Tablespace Page size (bytes)	= 4096
Tablespace Extent size (pages)	= 4
Automatic Prefetch size enabled	= Yes
Buffer pool ID currently in use	= 1
Buffer pool ID next startup	= 1
Using automatic storage	= Yes
Auto-resize enabled	= Yes
File system caching	= Yes
Tablespace State	= 0x'00000000'
Detailed explanation:	
Normal	
Tablespace Prefetch size (pages)	= 4
Total number of pages	= 16384
Number of usable pages	= 16380
Number of used pages	= 8868
Number of pending free pages	= 0
Number of free pages	= 7512
High water mark (pages)	= 8868
Initial tablespace size (bytes)	= 33554432

Current tablespace size (bytes)	= 67108864
Maximum tablespace size (bytes)	= NONE
Increase size (bytes)	= AUTOMATIC
Time of last successful resize	=
Last resize attempt failed	= No
Rebalancer Mode	= No Rebalancing
Minimum Recovery Time	=
Number of quiescers	= 0
Number of containers	= 1

Note that Using *automatic storage* = Yes and *Increase size (bytes)* = AUTOMATIC are shown in the snapshot.

Using the SNAPCONTAINER view

- Use the SYSIBMADM.SNAPCONTAINER view to retrieve container information

```
SELECT SNAPSHOT_TIMESTAMP, TBSP_NAME, TBSP_ID,
CONTAINER_NAME, CONTAINER_ID, CONTAINER_TYPE,
ACCESSIBLE, TOTAL_PAGES, USABLE_PAGES,
DBPARTITIONNUM FROM SYSIBMADM.SNAPCONTAINER ORDER
BY DBPARTITIONNUM
```

TBSP_NAME	TBSP_ID	CONTAINER_NAME	CONTAINER_ID	CONTAINER_TYPE	TOTAL_PAGES	USABLE_PAGES
SYSCATSPACE	0	C:\DB2\NODE0000\W...	0	FILE_EXTENT_TAG	8192	8188
TEMPSPACE1	1	C:\DB2\NODE0000\W...	0	PATH	0	0
USERSPACE1	2	C:\DB2\NODE0000\W...	0	FILE_EXTENT_TAG	8192	8160
SYSTOOLSPACE	3	C:\DB2\NODE0000\W...	0	FILE_EXTENT_TAG	8192	8188
DMS01	4	C:\DB2\dmss\dmss01	0	FILE_EXTENT_TAG	106	100
DMS02	5	C:\DB2\dmss\dmss02	0	FILE_EXTENT_TAG	14	12
DMS03	6	C:\DB2\dmss\dmss03	0	FILE_EXTENT_TAG	728	720
DMS04	7	C:\DB2\dmss\dmss04	0	FILE_EXTENT_TAG	22	20
DMS05	8	C:\DB2\dmss\dmss05	0	FILE_EXTENT_TAG	16	14
DMS06	9	C:\DB2\dmss\dmss06	0	FILE_EXTENT_TAG	40	36
SMS01	10	C:\DB2\smss\smss01	0	PATH	0	0

© Copyright IBM Corporation 2007

Figure 4-25. Using the SNAPCONTAINER view

CF238.3

Notes:

This administrative view allows you to retrieve tablespace_container logical data group snapshot information for the currently connected database.

Used with the SNAPTbsp, SNAPTbsp_Part, SNAPTbsp_Quiescer and SNAPTbsp_Range administrative views, the SNAPCONTAINER administrative view returns data equivalent to the GET SNAPSHOT FOR TABLESPACES ON database-alias CLP command.

Alter Table Space Wizard

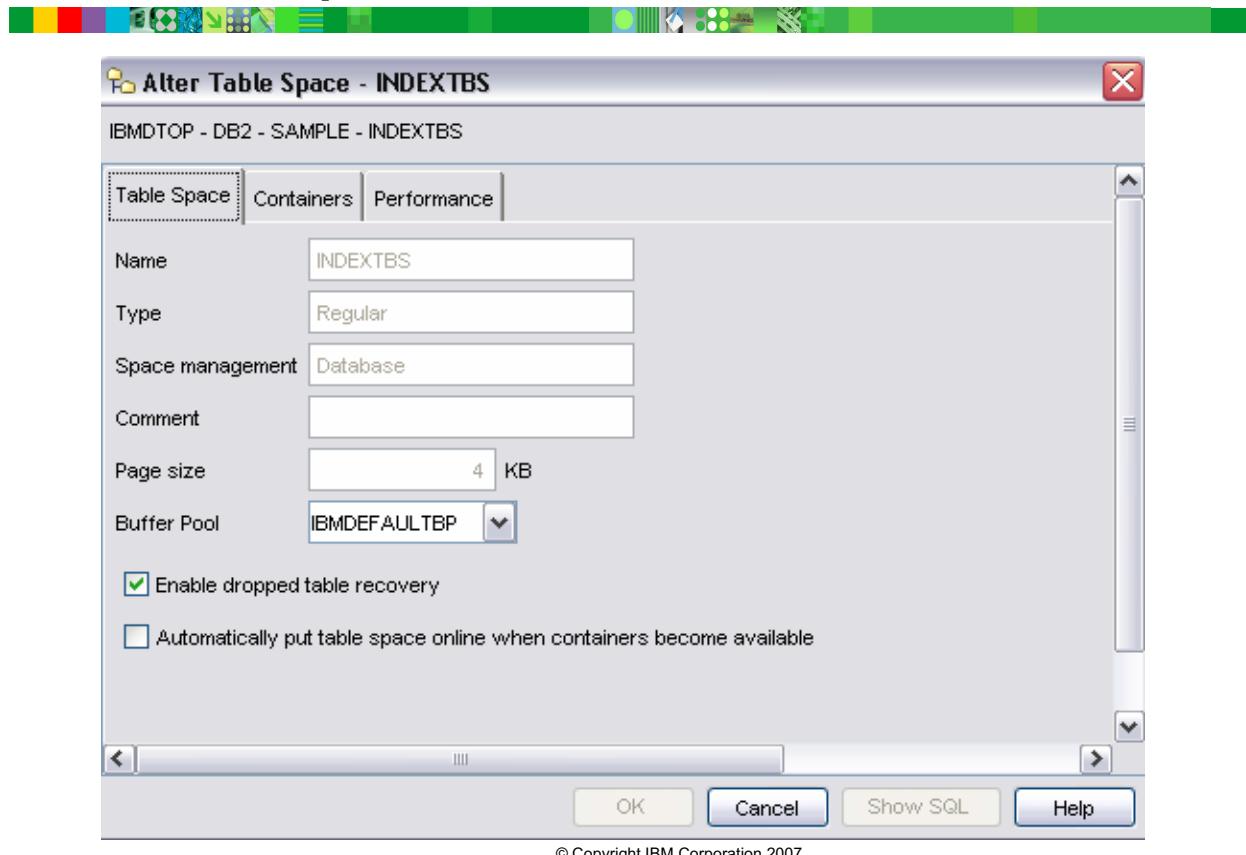


Figure 4-26. Alter Table Space Wizard

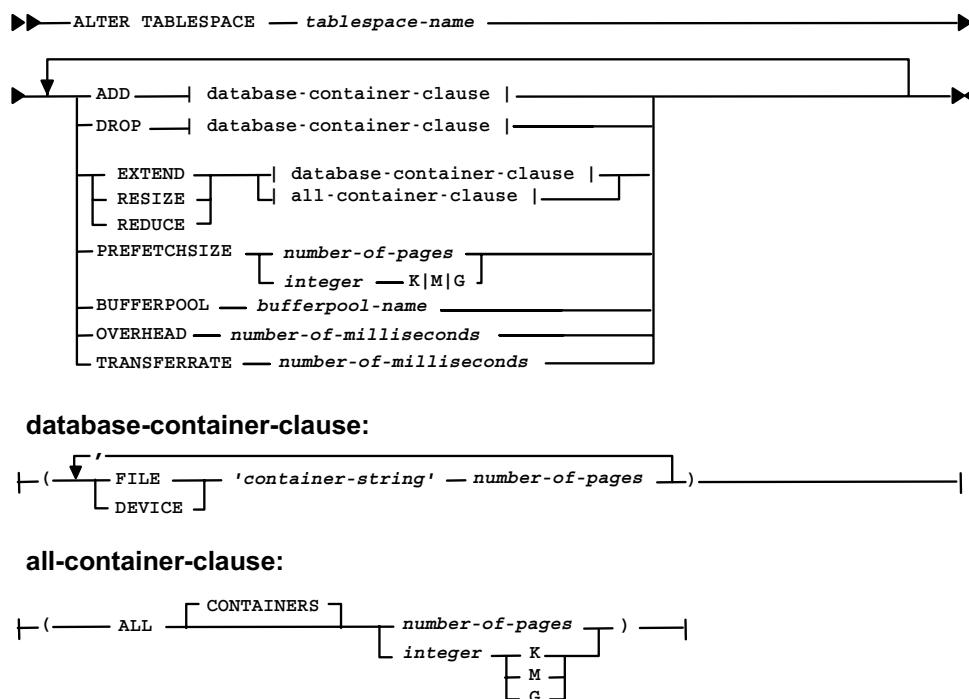
CF238.3

Notes:

What can be altered from the Alter Tablespace GUI?

- On **Table Space** tab:
 - Comment for table space
 - Buffer pool assignment
 - Setting for dropped table recovery
 - Setting for bringing table space online when containers are available
- On **Containers** tab:
 - Add a container to a DMS table space
 - Drop a container from a DMS table space
 - Resize a DMS table space container
 - Manage stripe sets
- On **Performance** tab:
 - Prefetch size
 - Overhead
 - Transfer rate

ALTER TABLESPACE syntax



© Copyright IBM Corporation 2007

Figure 4-27. ALTER TABLESPACE syntax

CF238.3

Notes:

The ALTER TABLESPACE statement is used to do the following:

- Add or drop a container in a DMS table space
- Extend, reduce, or resize DMS table space containers
- Modify the PREFETCHSIZE setting for a table space
- Modify the OVERHEAD setting for a table space
- Modify the TRANSFERRATE setting for a table space
- Modify the BUFFERPOOL assigned for a table space

When adding or removing space from the table space, the following rules must be followed:

- EXTEND and RESIZE can be used in the same statement, provided that the size of each container is increasing.
- REDUCE and RESIZE can be used in the same statement, provided that the size of each container is decreasing.
- EXTEND and REDUCE cannot be used in the same statement, unless they are being directed to different partitions.

- ADD cannot be used with REDUCE or DROP in the same statement, unless they are being directed to different partitions.
- DROP cannot be used with EXTEND or ADD in the same statement, unless they are being directed to different partitions.

ADD

Specifies that a new container is to be added to the table space. Once the new container has been added and the transaction is committed, the contents of the table space are automatically rebalanced across the containers. Access to the table space is not restricted during the rebalancing.

If you are adding more than one container to a table space, they should be added at the same time, in the same ALTER TABLESPACE statement, so that the cost of rebalancing is incurred only once.

DROP

Specifies that one or more containers are to be dropped from the table space.

EXTEND

Specifies that existing containers are being increased in size. The size specified is the size by which the existing container is increased. If the all-containers-clause is specified, then all containers on the table space will increase by this size.

REDUCE

Specifies that existing containers are to be reduced in size. The size specified is the size by which the existing container is decreased. If the all-containers-clause is specified, all containers in the tablespace will decrease by this size.

RESIZE

Specifies that the size of existing containers is being changed. The size specified is the new size for the container. If the all-containers-clause is specified, then all containers in the table space will be changed to this size. If the operation affects more than one container, these containers must all either increase in size, or decrease in size. It is not possible to increase some while decreasing others.

PREFETCHSIZE

Specifies the number of PAGESIZE pages that will be read from the table space when data prefetching is being performed.

BUFFERPOOL

Specifies the name of the buffer pool to be used for tables in this table space.

OVERHEAD

A number that specifies the I/O controller overhead and disk seek and latency time, in milliseconds. The number should be an average for all containers that belong to the table space, if not the same for all containers. This value is used to determine the cost of I/O during query optimization.

TRANSFERRATE

A number that specifies the time to read one page into memory, in milliseconds. The number should be an average for all containers that belong to the table space, if not the same for all containers. This value is used to determine the cost of I/O during query optimization.



Important

Caution: When using RESIZE or EXTEND with device containers, ensure that you do not overallocate the space in the underlying device.

The RENAME TABLESPACE statement allows you to rename an existing table space.

You can turn AUTORESIZE off by specifying AUTORESIZE NO in the ALTER TABLESPACE command.

This specifies that the auto-resize capability of a DMS table space or an automatic storage table space is to be disabled. If the auto-resize capability is disabled, any values that have been previously specified for INCREASESIZE or MAXSIZE will not be kept.

Placement considerations

- 
- If index scans are required:
 - Consider separate table spaces for index and data
- Consider placing index table space on fastest media available
- Consider placing entire table onto separate table space
 - If accessed frequently
- Consider EXTENTSIZE
 - Trade-off between space (very small tables) and performance
- I/O Prefetch

© Copyright IBM Corporation 2007

Figure 4-28. Placement considerations

CF238.3

Notes:

Having the ability to logically group tables in table spaces opens up many new possibilities for the database and system administrator. It also creates the necessity of planning to be done before the database creation is done, and to determine the optimal table space configuration. One question to ask is, “Where will I keep my catalogs, data objects, and indexes?”

Consider, as in our example, that you will have a very large database. We will keep employee resumes and pictures in our database; possibly these are types of data we had not considered storing in our database in the past. For performance, we chose to separate the indexes onto a separate table space (and disk). Multiple queries can read data and indexes in parallel, searching for the index and table data at the same time. However, for a single query, the database manager must read the index before reading the data. The performance benefit in this case is the reduction of the read/write head movement that would occur when index and data I/O requests go to the same disk.

Consider placing a table onto a separate table space if that table will be accessed frequently. You may be able to control the placement onto disk for that table. For example,

placing a frequently accessed table closer to the middle of the disk on a UNIX platform could ensure faster access.

The EXTENTSIZE parameter is the number of pages written before writing continues in another extent or segment. The EXTENTSIZE is a trade-off between the potential for wasting space and the cost of extending a database object. If you choose a small EXTENTSIZE (for example, two 4 KB pages or 8 KB), then every time an object grows by 8 KB, you will need to add a new extent. On the other hand, if you choose a large EXTENTSIZE (for example, 256 4 KB pages or 1024 pages), then there can be many unused pages in an extent that contains a small table or the last extent of a large table.

The objective of I/O prefetch is to improve the performance of queries that need a large amount of I/O. This is done by creating a separate process that actually performs the reads of large amounts of data.

4.4 Describe and modify the Database Configuration

Database Configuration

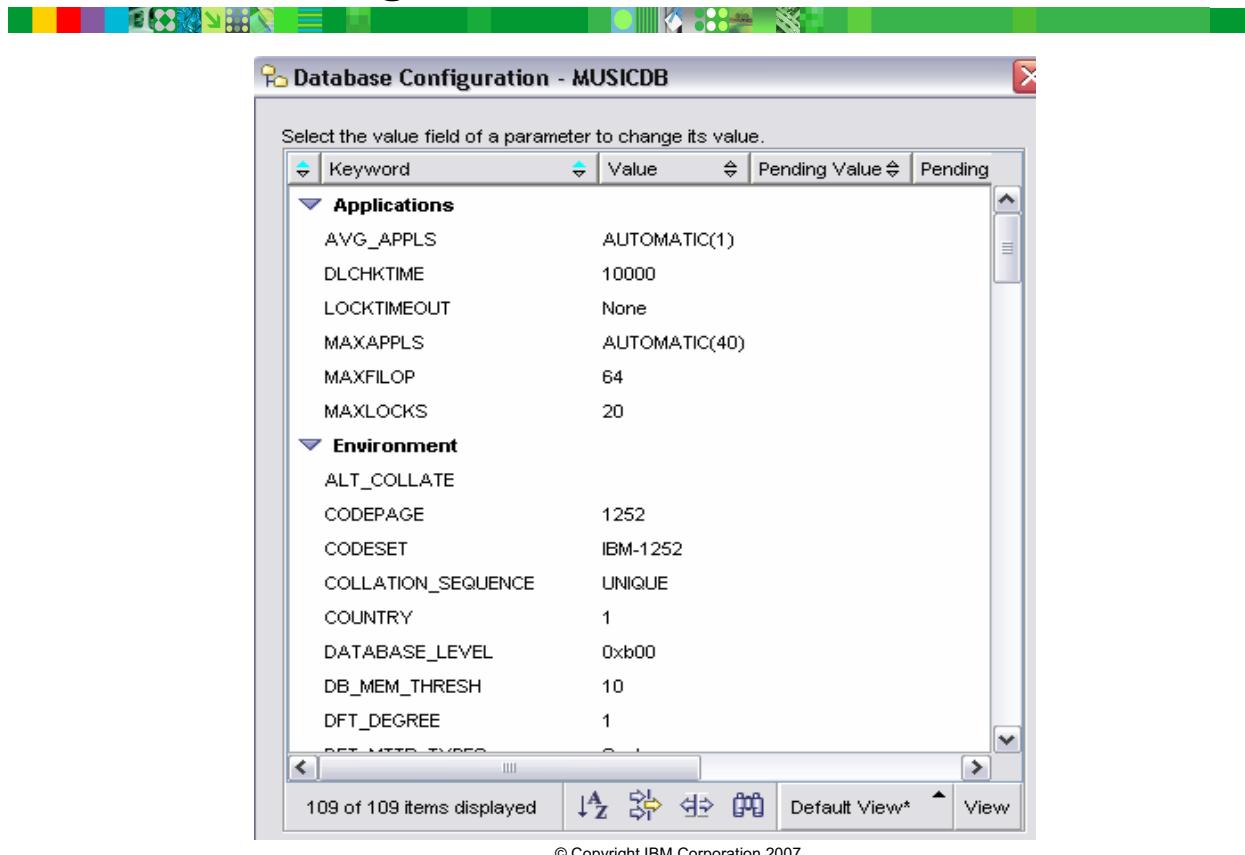


Figure 4-29. Database Configuration

CF238.3

Notes:

Use the Database Configuration window to view or change the database configuration parameter values. The database configuration parameters (**Keyword**) and their values for this database are displayed. You can use the options available from **View** to rearrange the display.

The database configuration file is created when a DB2 database is created. The parameters it contains affect resources at the database level. Values for many of these parameters can be changed from the default values to improve performance.

CLP or the Control Center may be used to get a listing of the database configuration file. The GET DB CFG command will list the parameters contained in the database configuration file.

The updatable parameters in the database configuration file can be changed using the UPDATE DB CFG command, or via the database configuration GUI.

The database territory, code set, country code, and code page are recorded in the database configuration file. However, these parameters cannot be changed.

Database Manager Configuration

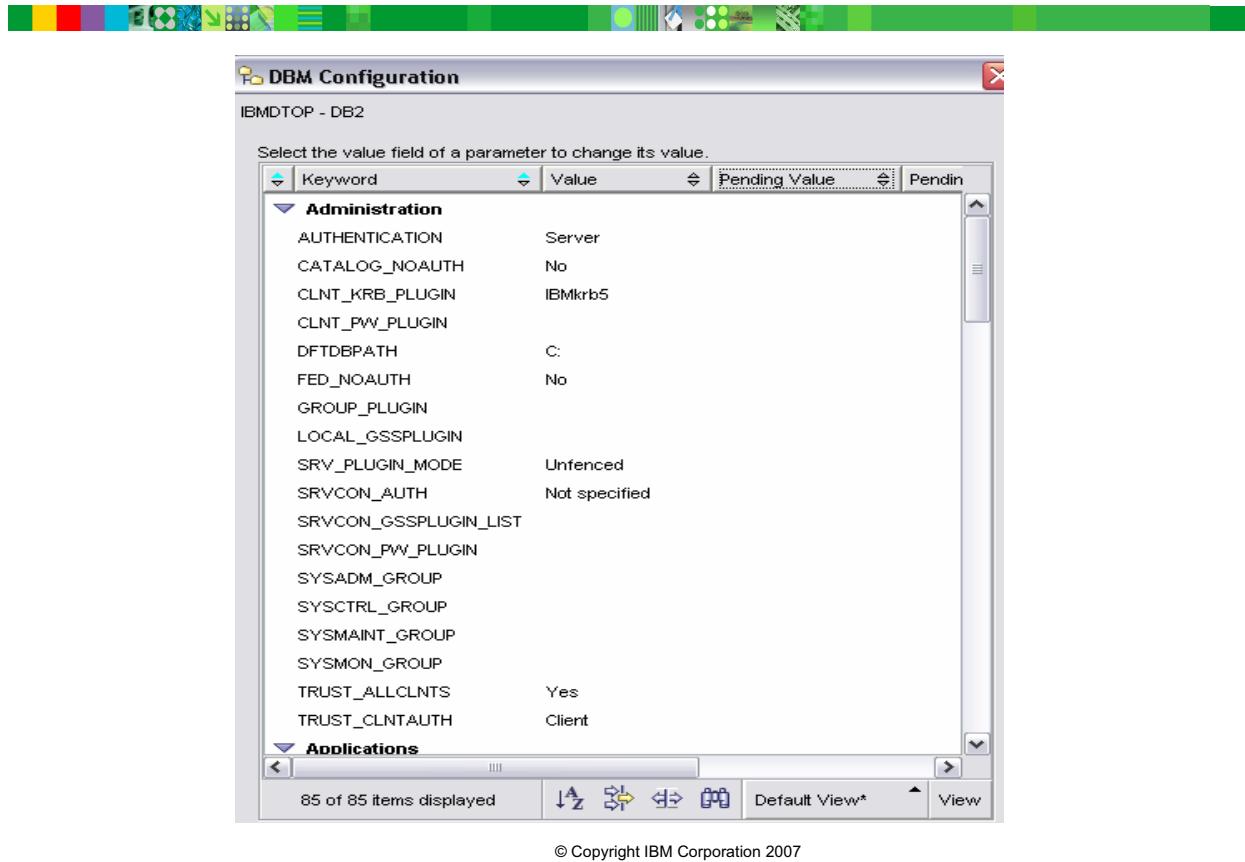


Figure 4-30. Database Manager Configuration

CF238.3

Notes:

Use the DBM Configuration window to view or change the database manager configuration parameter values.

The database manager configuration file is created when a DB2 instance is created. The parameters it contains affect system resources at the instance level, independently of any one database that is part of that instance. Values for many of these parameters can be changed from the system default values to improve performance or increase capacity, depending on your system's configuration.

The database manager configuration variables (**Keyword**) and their values for this instance are displayed. You can use the options available from **View** to rearrange the display.

Every instance has a database manager configuration file, specifying global parameters that affect the instance. The GET DBM CFG command can be used to list the database manager configuration file.

Values of updatable parameters can be changed using the UPDATE DBM CFG command.

Database directories overview

- db2 create database ourdb

```
db2 list db directory
System Database Directory
Number of entries in the directory = 2
Database 1 entry:
Database alias           = OURDB
Database name            = OURDB
Local database directory = C:
Database release level   = b.00
Comment                  =
Directory entry type     = Indirect
Catalog database partition number = 0
Alternate server hostname =
Alternate server port number =

Database 2 entry:
Database alias           = SAMPLE
Database name            = SAMPLE
Local database directory = C:
Database release level   = b.00
Comment                  = A sample database
Directory entry type     = Indirect
Catalog database partition number = 0
Alternate server hostname =
Alternate server port number =
```

db2 catalog db sample as testdb on c:
 DB20000I The CATALOG DATABASE command completed successfully.
 DB21056W Directory changes may not be effective until the directory cache is refreshed.

db2 uncatalog db testdb
 DB20000I The UNCATALOG DATABASE command completed successfully.
 DB21056W Directory changes may not be effective until the directory cache is refreshed.

Figure 4-31. Database directories overview

CF238.3

Notes:

The database directory maintains a list of databases that can be connected to.

Either CLP or the Control Center may be used to change directory entries. The Control Center allows a change to be made without uncataloging and recataloging the directory. There is no CHANGE DIRECTORY CLP command. If you wish to change a catalog entry using CLP, you must uncatalog the directory entry and then recatalog the entry using the CATALOG command.

The database alias must be unique. The database name does not have to be unique. A reason to catalog a local database is to change its alias, which is the name by which users and programs identify the database. When a database is created, its alias is the same as its name, unless specified otherwise on the CREATE DATABASE command. The name by which users and programs refer to a database can be changed without having to DROP and re-CREATE the database.

You may want a database catalogued with more than one alias name.

Checkpoint

Exercise — Unit Checkpoint

- ___ 1. Name the two types of table spaces:

- ___ 2. Which type of table space must be used if you want to store the index data separately from the table data?
- ___ 3. Which type of table space should be used to allow the operating system to allocate pages to the table space as needed?

Unit summary



Having completed this unit, you should be able to:

- Review specifics of creating a database
- Explore the System Catalog tables and views
- Compare DMS versus SMS table spaces
- Describe how to setup and manage a DB2 database with Automatic Storage enabled
- Differentiate between table spaces, containers, extents, and pages
- Define table spaces
- Use the **get snapshot for tablespaces** command to display table space statistics
- Explore database configuration parameters

© Copyright IBM Corporation 2007

Figure 4-32. Unit summary

CF238.3

Notes:

Unit 5. Creating database objects

What this unit is about

This unit provides information about DB2's physical structure (files and directories), tables, views, aliases, referential integrity, check constraints, triggers, and LOBs.

Database objects can be created, updated, and listed with the Command Line Processor (CLP) or the Control Center graphical tool.

What you should be able to do

After completing this unit, you should be able to:

- List DB2 object hierarchy and physical directories and files
- Create the following objects:
 - Schema, Table, View, Alias, Index
 - Explore the use of table partitioning
 - Review the use of Temporary Tables
 - Explore the use and implementation of Check Constraints, Referential Integrity and Triggers
 - Exploring the need for and the use of Large Objects
 - Recognize XML and its native store as critical infrastructure for emerging technologies

How you will check your progress

Accountability:

- Machine exercises

References

*Command Reference
Administration Guide*

Unit objectives



After completing this unit, you should be able to:

- List DB2 object hierarchy and physical directories and files
- Learn to create the following objects:
 - Schema, Table, View, Alias, Index
- Explore the use of table partitioning
- Review the use of Temporary Tables
- Explore the use and implementation of Check Constraints, Referential Integrity and Triggers
- Exploring Large Objects: the need and usage
- Study an overview of XML as a native store

© Copyright IBM Corporation 2007

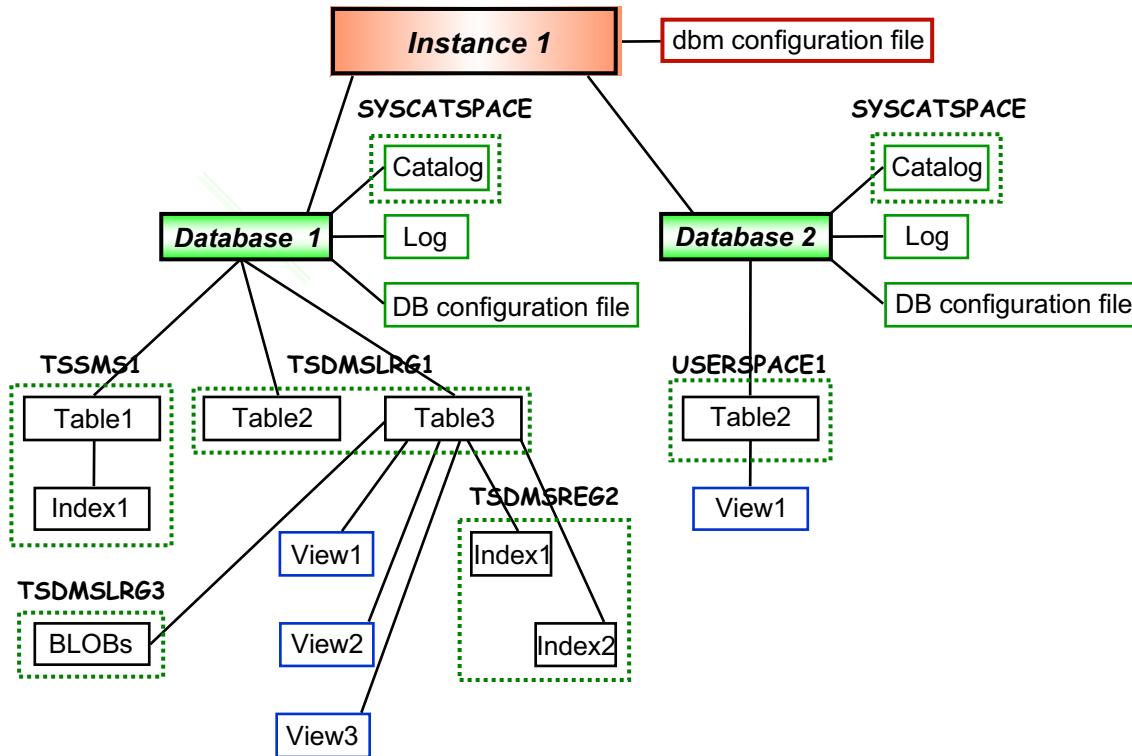
Figure 5-1. Unit objectives

CF238.3

Notes:

5.1 DB2 object hierarchy and physical directories and files

DB2 object hierarchy



© Copyright IBM Corporation 2007

Figure 5-2. DB2 object hierarchy

CF238.3

Notes:

Each DB2 instance has its own database manager configuration file. Its global parameters affect the system resources allocated to DB2 for an individual instance. Its parameters may be changed from the system default values to improve performance or increase capacity, depending on the workstation configuration.

Each instance may have multiple databases. A relational database presents data as a collection of tables. A table consists of a defined number of columns and any number of rows. Each database includes a set of system catalog tables, which describe the logical and physical structure of the data (like a table or view), or contain statistics of the data distribution; a configuration file containing the parameter values allocated for the database; and a recovery log with ongoing transactions and archiveable transactions.

Each table may have multiple indexes. Indexes may provide a faster way to access table data. Each table may have multiple views. Views may be associated with more than one base table.

The physical objects in a database are assigned to table spaces. When creating a table, you can decide to have certain objects such as indexes and large object (LOB) data kept separately from the rest of the table data. By default, all objects referencing a table reside in the same table space where the table itself resides. A table space can also be spread over one or more physical storage devices.

In the visual, two databases are shown.

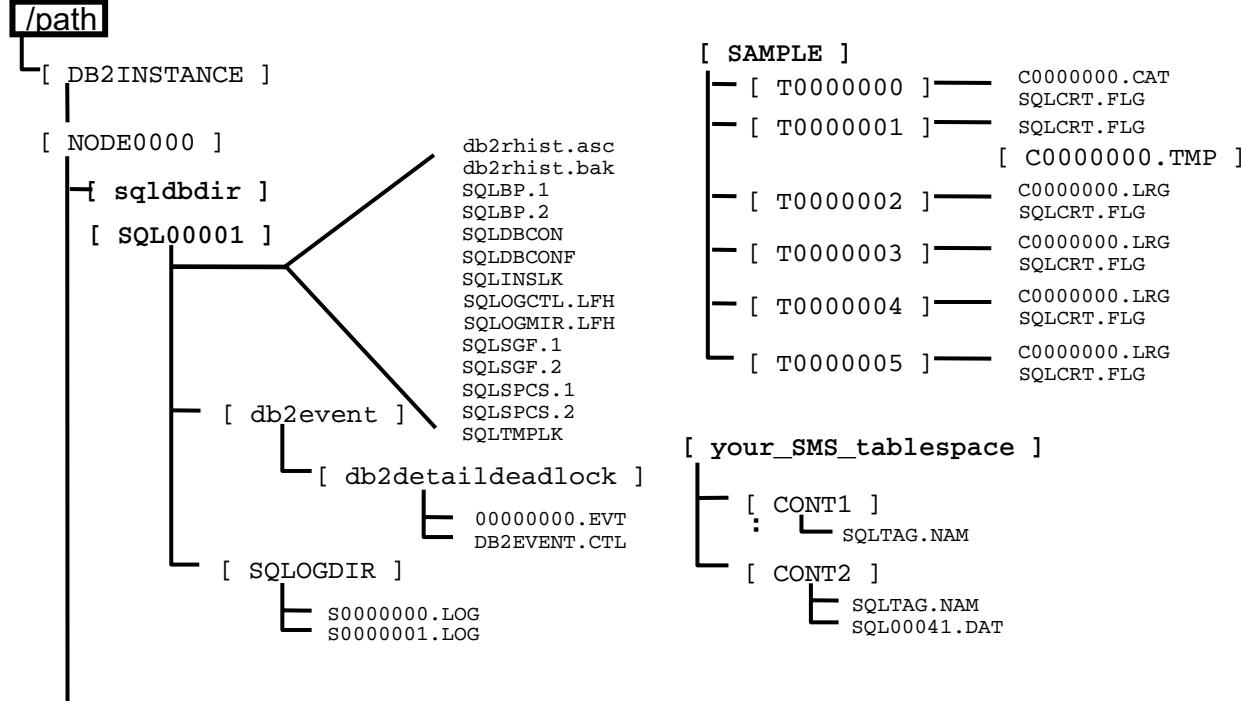
For Database 1:

- The system catalog tables are in table space SYSCATSPACE.
- Table 1 and its one Index are in a SMS table space named TSSMS1.
- Tables 2 and Table 3 are both assigned to the Large DMS table space TSDMSLRG1.
- Two Indexes for Table 3 are assigned to the Regular table space TSDMSREG2.
- The Large Object data columns from Table 3 are assigned to the Large table space TSDMSLRG3.

For Database 2:

- The system catalog tables are in table space SYSCATSPACE.
- Table 2 is assigned to the table space USERSPACE1. By default, USERSPACE1 would be an Automatic Storage managed table space.

Database physical directories and files



© Copyright IBM Corporation 2007

Figure 5-3. Database physical directories and files

CF238.3

Notes:

Do not directly edit or alter these files. Do not move these files. The only supported means of database access is through the use of DB2 administration tools, such as the Control Center.

When an instance is created, the following directory structure is created:

```
$HOME/sqllib (UNIX)
x:\install_location\${DB2INSTANCE}\SQLLIB (Windows)
```

After you create the first database for an instance, the following directory structure exists:

```
/path/${DB2INSTANCE}/NODE0000/sqldbdir/sqldbdir
/path/${DB2INSTANCE}/NODE0000/SQL00001
```

When a database is created, a local database directory is created. The name of the file is **sqldbdir**.

When a database is created, the database manager creates a separate subdirectory to store the database objects associated with that database. The naming scheme used by the database manager is /path/\${DB2INSTANCE}/NODE0000/SQL00001, where SQL00001

contains the objects associated with the first database created. On the CREATE DATABASE command a path (for example, /path) is specified. The database manager maintains these subdirectories automatically. You can determine which subdirectory is related to each database by using the command LIST DB DIRECTORY or LIST DB DIRECTORY ON <path>.

The database directory contains the following files that are created as part of the CREATE DATABASE command.

- The files SQLBP.1 and SQLBP.2 contain buffer pool information. Each file has a duplicate copy to provide a backup.
- The files SQLSPCS.1 and SQLSPCS.2 contain table space information. Each file has a duplicate copy to provide a backup.
- The files SQLSGF.1 and SQLSGF.2 contain storage path information associated with the database's automatic storage. Each file has a duplicate copy to provide a backup.
- The SQLDBCON file contains database configuration information. Do not edit this file. To change configuration parameters, use either the Control Center or the command-line statements UPDATE DATABASE CONFIGURATION and RESET DATABASE CONFIGURATION.
- The DB2RHIST.ASC history file and its backup DB2RHIST.BAK contain history information about backups, restores, loading of tables, reorganization of tables, altering of a table space, and other changes to a database.
 - The contents of the DB history file can be viewed using the DB2 Journal or by using the LIST HISTORY command.
- The DB2TSCHNG.HIS file contains a history of table space changes at a log-file level. For each log file, DB2TSCHNG.HIS contains information that helps to identify which table spaces are affected by the log file. Table space recovery uses information from this file to determine which log files to process during table space recovery. You can examine the contents of both history files in a text editor.
- The log control files, SQLOGCTL.LFH and SQLOGMIR.LFH, contain information about the active logs.
 - Recovery processing uses information from this file to determine how far back in the logs to begin recovery. The SQLOGDIR subdirectory contains the actual log files.
- The SQLINSLK file helps to ensure that a database is used by only one instance of the database manager.

SMS table spaces

In SMS table spaces, tables are stored in files at the operating system level. The first table data file (**SQL00001.DAT**) is created in a container (segment), for example SQLT0000.0. DB2 will extend the DAT file one extent at a time, using the extent size for the table space (defaults to **DFT_EXTENT_SZ** if not explicitly indicated). The next extent of data would be written to the SQL00001.DAT in the next container, for example SQLT0000.1, if more than

one container has been defined for the table space. The same mechanism is followed for INX, LF, LOB, and LBA files.

Each subdirectory or container has a file created in it called SQLTAG.NAM. This file marks the subdirectory as being in use so that subsequent table space creation does not attempt to otherwise use these subdirectories.

In addition, a file called SQL*.DAT stores information about each table that the subdirectory or container contains. The asterisk (*) is replaced by a unique set of digits that identifies each table. For each SQL*.DAT file there might be one or more of the following files, depending on the table type, the reorganization status of the table, or whether indexes, LOB, or LONG fields exist for the table:

- SQL*.BKM (contains block allocation information if it is an MDC table)
- SQL*.LF (contains LONG VARCHAR or LONG VARGRAPHIC data)
- SQL*.LB (contains BLOB, CLOB, or DBCLOB data)
- SQL*.XDA (contains XML data)
- SQL*.LBA (contains allocation and free space information about SQL*.LB files)
- SQL*.INX (contains index table data)
- SQL*.IN1 (contains index table data)
- SQL*.DTR (contains temporary data for a reorganization of an SQL*.DAT file)
- SQL*.LFR (contains temporary data for a reorganization of an SQL*.LF file)
- SQL*.RLB (contains temporary data for a reorganization of an SQL*.LB file)
- SQL*.RBA (contains temporary data for a reorganization of an SQL*.LBA file)

On the visual, you also see two user-defined table space directories:

- **SMS** directory, for the SMS table space, and the files it includes has the same structure as the SMS table spaces that were created under SQL00001. Note that .DAT or .INX or .LBA files could be found under this directory.
- **DMS** directory, for the DMS table space, the specified disk space is preallocated. Every DMS table space has its internal metadata that is used by DB2 to manage the disk space among the objects assigned to the tablespace.

Temporary transient files that are not shown on the visual include the following:

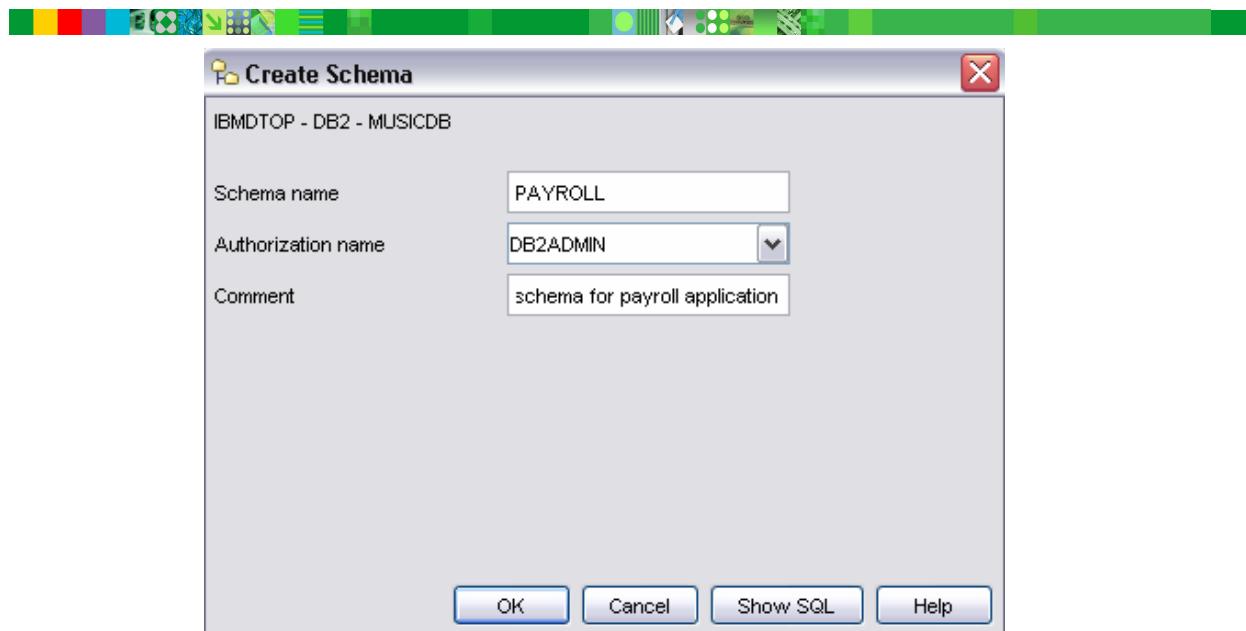
- **SQLnnnnn.DTR** is a temporary data file for a reorg of a DAT file.
- **SQLnnnnn.LFR** is a temporary data file for a reorg of a LF file.
- **SQLnnnnn.RLB** is a temporary data file for a reorg of a LB file.
- **SQLnnnnn.RBA** is a temporary data file for a reorg of a LBA file.

Temporary files used by REORG should never be erased; they may be the only place your data still exists if there is a failure in the midst of a REORG process.

SQLTMPLK and **SQLINSLK** are files used to help ensure that a database is only used by one instance of the database manager.

5.2 Creating objects

Create Schema



```
CREATE SCHEMA PAYROLL AUTHORIZATION DB2ADMIN;
COMMENT ON Schema PAYROLL IS 'schema for
payroll application';
```

© Copyright IBM Corporation 2007

Figure 5-4. Create Schema

CF238.3

Notes:

To access the Create Schema GUI from the Control Center:

- Select **Schemas** (right-click the DB node *Schemas*)
 - Select **Create**

The Create Schema GUI has the following:

- Schema name
- Authorization name
- Comment

The **Show SQL** or **Show Command** button is available on all Control Center dialogs that generate SQL or DB2 commands. You can display the SQL statement save it as a script.

When the schema is explicitly created with the CREATE SCHEMA statement, the schema owner is granted CREATEIN, DROPIN, ALTERIN privileges on the schema with the ability to grant these privileges to other users.

A schema name or authorization name cannot begin with **SYS**.

While organizing your data into tables, it may also be beneficial to group tables (and other related objects) together. This is done by defining a schema. Information about the schema is kept in the system catalog tables of the database to which you are connected. As other objects are created, they can be placed within this schema.

An authorization ID that holds SYSADM or DBADM authority can create a schema with any valid schema name or authorization name. Any ID can explicitly create a schema that matches the authorization ID of the statement.

When the schema is explicitly created with the CREATE SCHEMA statement, the schema owner is granted CREATEIN, DROPIN, ALTERIN, GRANTIN privileges on the schema with ability to grant to other users.

You can create a schema and include certain SQL statements with it (CREATE TABLE, excluding typed tables and materialized query tables; CREATE VIEW statement, excluding typed views; CREATE INDEX statement; COMMENT statement; GRANT statement). For example, the following is a *single* statement:

```
CREATE SCHEMA pers
CREATE TABLE ORG (
    deptnumb SMALLINT NOT NULL,
    deptname VARCHAR(14),
    manager SMALLINT,
    division VARCHAR(10),
    location VARCHAR(13),
    CONSTRAINT pkeydno PRIMARY KEY (deptnumb),
    CONSTRAINT fkeymgr FOREIGN KEY (manager)
        REFERENCES staff (id)
)
CREATE TABLE STAFF (
    id SMALLINT NOT NULL,
    name VARCHAR(9),
    dept SMALLINT,
    job VARCHAR(5),
    years SMALLINT,
    salary DECIMAL(7,2),
    comm DECIMAL(7,2),
    CONSTRAINT pkeyid PRIMARY KEY (id),
    CONSTRAINT fkeydno FOREIGN KEY (dept)
        REFERENCES org (deptnumb)
)
```

Thus, you can use a single statement to create two tables that are dependent on each other, rather than having to create the first with primary key, the second with primary and foreign key, and then alter the first to add foreign key.

Unqualified object names in any SQL statement within the CREATE SCHEMA statement are implicitly qualified by the name of the created schema.

Set current schema



```
connect to musicdb user Keith;
select * from employee;
■ will select from KEITH.EMPLOYEE
```

```
set current schema = 'PAYROLL';
select * from employee;
■ will select from PAYROLL.EMPLOYEE
```

© Copyright IBM Corporation 2007

Figure 5-5. Set current schema

CF238.3

Notes:

When accessing data within DB2, unqualified references will be implicitly qualified with the authorization ID that was used to connect to the database. You can override this by setting the CURRENT SCHEMA. The initial value of the CURRENT SCHEMA special register is equivalent to USER.

The example on the graphic shows that a user *KEITH* is connecting to the database. If Keith issues a select against the EMPLOYEE table, the table that will be accessed will be KEITH.EMPLOYEE. If he sets his current schema to *PAYROLL*, then a select against the EMPLOYEE table will be directed against the PAYROLL.EMPLOYEE table.

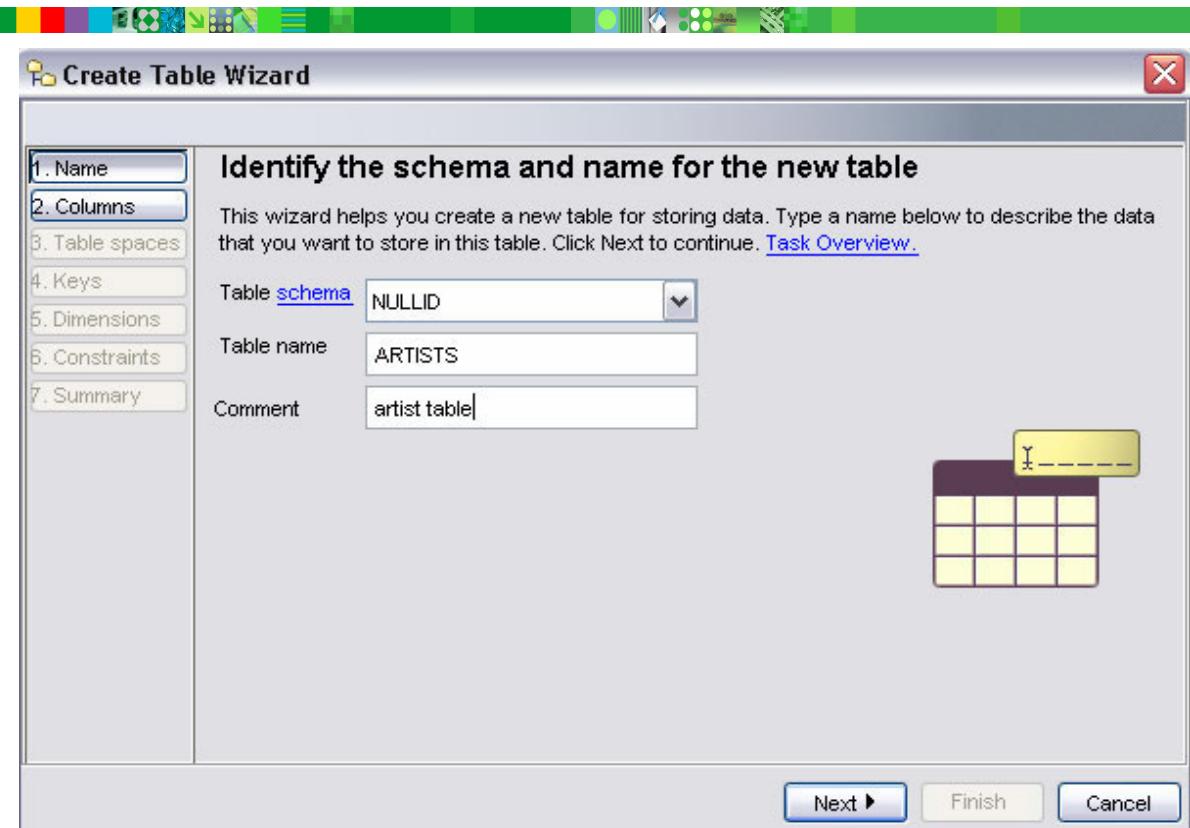
Alternative syntax includes:

```
SET CURRENT SCHEMA = 'PAYROLL'
SET SCHEMA 'PAYROLL'
SET CURRENT SQLID 'PAYROLL'
```

Note that the use of the = is optional in all of these statements.

The value of the CURRENT SCHEMA special register is used as the schema name in all dynamic SQL statements where an unqualified reference to a database object exists.

Create Table



© Copyright IBM Corporation 2007

Figure 5-6. Create Table

CF238.3

Notes:

To access the Create Table GUI from the Control Center:

- Select **Tables** (right-click the DB node *Tables*)
 - Select **Create**

The Create Table Wizard has the following pages:

- **Name**

On this page, you specify the schema and name for the table.

- **Columns**

This page displays the columns of the table. On this page, you can add, edit, remove, or rearrange the columns. You can also choose from a list of predefined columns and add them to the table.

- **Table Space**

Use this page to specify the table space that will store the data. On this page, you can also launch the Create Table Space wizard if you need to create a table space.

- **Keys**

Use this page to define the primary, unique, and foreign keys for the table. You can add, change, or remove keys. If you are creating the table on a partitioned database, then this page allows you to define the partitioning key.

- A primary key is a column or set of columns that is used to uniquely identify each row on a table. It is used by the database to increase the efficiency of table operations. It is also used in the creation of foreign keys. A table cannot have more than one primary key, and no column of the primary key can contain a NULL value.
- A unique key is a key that is constrained so that no two of its values are equal. The columns of a unique key can contain one NULL value.
- A foreign key is a column or set of columns whose values are required to match at least one primary key value in another table.
- A partitioning key is a key that is part of the definition of a table in a partitioned database. The partitioning key is used to determine the partition on which the row of data is stored. If a partitioning key is defined, unique keys and primary keys must include the same columns as the partitioning key (they may have more columns). A table cannot have more than one partitioning key.

- **Dimensions**

If you are creating a multidimensional clustered table, then use this page to define the dimensions for the table. This is your only opportunity to define the table dimensions. The only way to change the dimensions for a table is to redefine the table and move the data into the new table.

- **Constraints**

This page lists the check constraints for the table, and allows you to add, change, or remove a check constraint. A check constraint sets restrictions on the data in a column that can be added to the table, and is enforced whenever the table has rows inserted or updated.

- **Summary**

This page allows you to review the decisions you have made for the table and to view the command that will create the table.

The items under these tabs will be covered later in this unit.

Create Table statement



```

connect to musicdb;
create table artists
(artno          smallint not null,
 name           varchar(50) with default 'abc',
 classification char(1) not null,
 bio            clob(100K) logged,
 picture        blob( 10M) not logged compact)

in dms01

index in dms02

long  in dms03;

```

© Copyright IBM Corporation 2007

Figure 5-7. Create Table statement

CF238.3

Notes:

A table consists of data logically arranged in columns and rows. DB2 supports page sizes of 4, 8, 16, and 32 KB. The number of columns, maximum row length, and maximum table size vary by page size. Regular tablespaces use a 4-byte Row ID, which allows 16 million pages with up to 255 rows per page. Non-temporary SMS-managed table spaces are all Regular table spaces. DMS-managed table spaces can be either Regular or Large table spaces. Large table spaces use a 6-byte Row ID, which allows up to 512 million pages to be addressed.

A table with a 4K page could be as large as 64 GB in a regular table space or 2 Terabytes in a Large table space. Using a 32K page size would allow a table in a Regular table space to be as large as 512GB or 16TB in a Large table space.

To create a table, you must be connected to a database, either implicitly or explicitly. You must have SYSADM or DBADM authority, or, you must have CREATETAB privilege and one of either IMPLICIT_SCHEMA or CREATEIN privilege on the schema. You must also have USE privilege over all table spaces referenced.

Tables are created using the SQL statement CREATE TABLE.

If no schema name is supplied with the table name, the value of the CURRENT SCHEMA special register is used as the schema name.

A table, view, or alias may be from one to 128 characters in length. A column name may be from one to 30 characters in length. An index name may be from one to 18 characters in length.

The data types may be INTEGER, INT, SMALLINT, BIGINT, DOUBLE, DOUBLE PRECISION, FLOAT, REAL, DECIMAL (precision-integer, scale-integer), DEC (precision-integer, scale-integer), NUMERIC (precision-integer, scale-integer), NUM (precision-integer, scale-integer), CHARACTER (integer), CHAR(integer), VARCHAR(integer), CHARACTER VARYING (integer), CHAR VARYING (integer), LONG VARCHAR, BLOB (integer), BINARY LARGE OBJECT (integer), CLOB (integer), CHARACTER LARGE OBJECT (integer), CHAR LARGE OBJECT (integer), DBCLOB (integer), GRAPHIC (integer), VARGRAPHIC (integer), LONG VARGRAPHIC, DATE, TIME, TIMESTAMP, DATALINK, or a user-defined distinct or structured type. More information about user-defined distinct types can be found in the appendix.

FOR BIT DATA specifies that the contents of the character or varying character column are to be treated as bit (binary) data.

- The following built-in function is available to use this type of column:
 - *GENERATE_UNIQUE()* which returns a unique value that can be used to provide unique values in a table. The type of unique value is CHAR(13) FOR BIT DATA.

NOT NULL prevents the column from containing null values. If you omit NOT NULL, the column can contain null values, and its default value is the null value.

NOT NULL WITH DEFAULT prevents a column from containing null values while allowing system defaults such as:

- 0 for a numeric data type
- Blanks for a fixed-length character string data type
- Current DATE, TIME, and TIMESTAMP for DATE, TIME, and TIMESTAMP data types (respectively).

WITH DEFAULT value allows a user-defined default value to be specified, whether or not NOT NULL is specified. This allows the table definer to select the specific default value for use when a column is not specified in an INSERT statement, or when DEFAULT is specified in an INSERT or UPDATE statement.

The CREATE TABLE and ALTER TABLE statements allow a constant value; special registers: CURRENT DATE, CURRENT TIME, CURRENT TIMESTAMP, CURRENT SCHEMA, or USER; or NULL to be specified following the DEFAULT keyword.

Referential integrity syntax is discussed later.

IN identifies the table space in which the table will be created. The table space must exist and be a REGULAR table space over which the authorization ID of the statement has USE privilege. If no other table space is specified, then all table parts will be stored in this table space.

If the **IN** clause is not specified, then if the table space IBMDEFAULTGROUP exists and it has sufficient page size, and if the user has USE privilege over the table space IBMDEFAULTGROUP, the table will be created in IBMDEFAULTGROUP. Otherwise, if a table space over which the user has USE privilege exists with sufficient page size, it will be chosen (if multiples exist, see next paragraph). If no table space exists that meets these criteria, the CREATE TABLE statement will fail.

If multiple table spaces exist that meet the page size and USE privilege requirements, the one with the smallest sufficient page size is selected. Preference is given according to who was granted USE privilege:

1. The authorization ID
2. A group to which the authorization ID belongs
3. PUBLIC

If more than one table space still qualifies, the final choice is made by the database manager.

INDEX IN identifies the table space in which any index on the table will be stored. This option is allowed only when the primary table space specified in the IN clause is a DMS table space. The specified table space must exist, and be a REGULAR or LARGE DMS table space over which the authorization ID of the statement has USE privilege.

Note that specifying which table space will contain a table's index can only be done when the table is created. The checking of USE privilege over the table space for the index is only carried out at table creation time. The database manager will not require that the authorization ID of a CREATE INDEX statement have USE privilege on the table space when an index is created later.

LONG IN identifies the table space in which the values of any long columns (LONG VARCHAR, LONG VARGRAPHIC or LOB data types) will be stored. This option is allowed only when the primary table specified in the IN clause is a DMS table space. The table space must exist and be a LARGE DMS table space over which the authorization ID of the statement has USE privilege.

A table may be created similar to an existing table by using the LIKE clause, for example: CREATE TABLE painters LIKE artists. The PAINTERS table will have the same columns (name and description) as the ARTISTS table.

The RENAME TABLE statement renames an existing table. Privileges needed to rename a table are SYSADM, DBADM authority or CONTROL privilege.

For more information, refer to the *SQL Reference* manual.

Table partitioning

- Data organization scheme in which table data is divided across multiple storage objects called data partitions or ranges
 - Each data partition is stored separately
 - These storage objects can be in different table spaces, in the same table space, or a combination of both
- Benefits
 - Easier roll-in and roll-out of table data
 - Supports very large tables
 - Use Hierarchical Storage Management (HSM) solutions more effectively
 - Easier administration
 - Flexible index placement
 - Better query processing

© Copyright IBM Corporation 2007

Figure 5-8. Table partitioning

CF238.3

Notes:

Storage objects behave much like individual tables, making it easy to accomplish fast roll-in by incorporating an existing table into a partitioned table using the ALTER TABLE...ATTACH statement. Likewise, easy roll-out is accomplished with the ALTER TABLE...DETACH statement. Query processing can also take advantage of the separation of the data to avoid scanning irrelevant data, resulting in better query performance for many data warehouse style queries.

Table data is partitioned as specified in the PARTITION BY clause of the CREATE TABLE statement. The columns used in this definition are referred to as the table partitioning key columns.

This organization scheme can be used in isolation or in combination with other organization schemes. By combining the DISTRIBUTE BY and PARTITION BY clauses of the CREATE TABLE statement, data can be spread across database partitions spanning multiple table spaces. The DB2 organization schemes include:

- DISTRIBUTE BY HASH

- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

Table partitioning functionality is available with DB2 Version 9.1 Enterprise Server Edition for Linux, UNIX, and Windows.

The following example creates a table **customer** where rows with `l_shipdate >= '01/01/2006'` and `l_shipdate <= '03/31/2006'` are stored in table space **ts1**, rows with `l_shipdate >= '04/01/2006'` and `l_shipdate <= '06/30/2006'` are in table space **ts2**, and so forth.

```
CREATE TABLE customer (l_shipdate, DATE, l_name CHAR(30))
    IN ts1, ts2, ts3, ts4, ts5
    PARTITION BY RANGE(l_shipdate) (STARTING FROM ('01/01/2006')
    ENDING AT ('12/31/2006') EVERY (3 MONTHS))
```

Large objects for a partitioned table reside, by default, in the same table space as the data. This default behavior can be overridden by using the LONG IN clause to specify one or more table spaces for the large objects. Create a table named **DOCUMENTS** whose large object data is to be stored (in a round-robin fashion for each data partition) in table spaces **TBSP1** and **TBSP2**.

```
CREATE TABLE documents
    id INTEGER,
    contents CLOB
    ) LONG IN tbsp1, tbsp2
    PARTITION BY RANGE (id) (STARTING 1 ENDING 1000 EVERY 100)
```

Alternatively, use the long form of the syntax to explicitly identify a large table space for each data partition. In this example, the CLOB data for the first data partition is placed in **TBSP3**, and the CLOB data for the remaining data partitions is spread across **TBSP1** and **TBSP2** in a round-robin fashion.

```
CREATE TABLE documents
    id INTEGER,
    contents CLOB
    ) LONG IN tbsp1, tbsp2
    PARTITION BY RANGE (id)
        (STARTING 1 ENDING 100 LONG IN TBSP3,
        STARTING 101 ENDING 1000 EVERY 100)
```

Create View

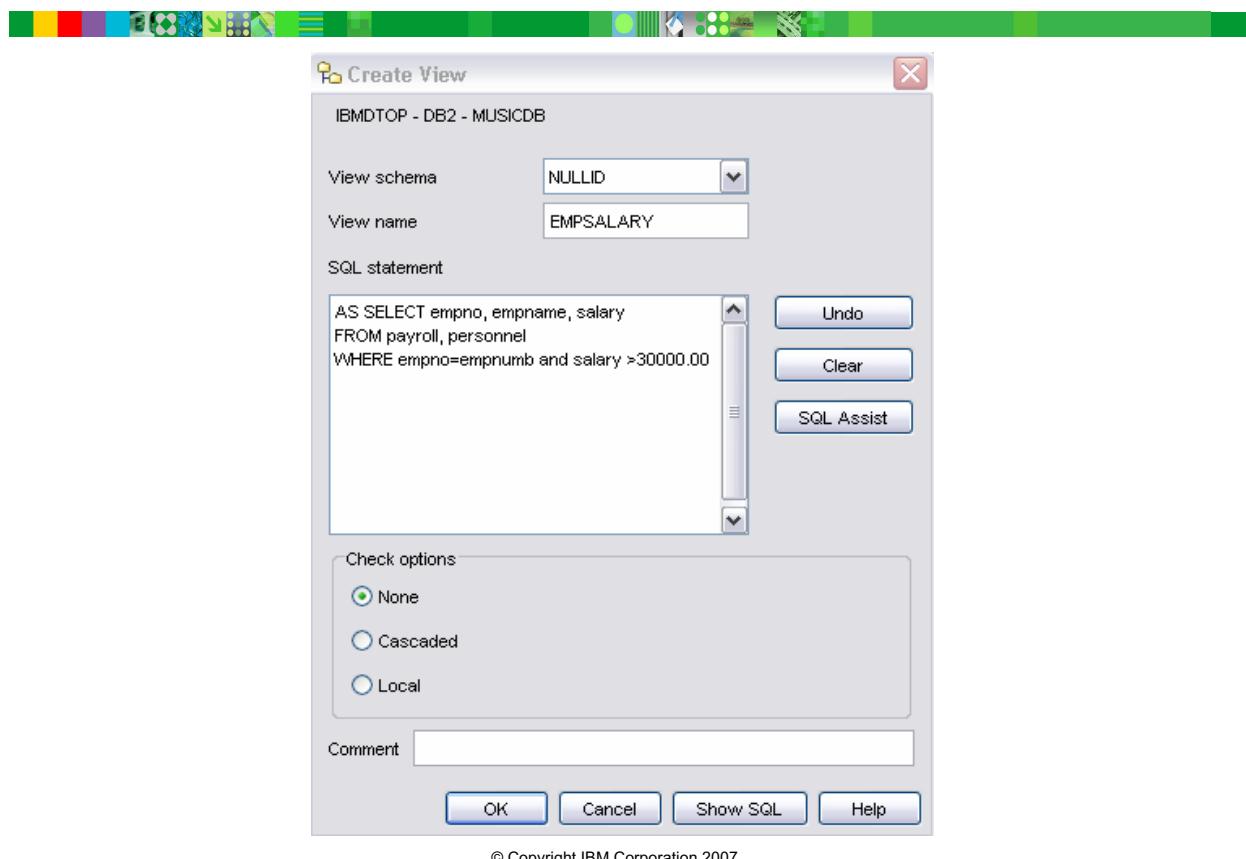


Figure 5-9. Create View

CF238.3

Notes:

To access the Create View GUI from the Control Center:

- Select **Views** (right-click the DB node *Views*)
 - Select **Create**

The Create Views GUI allows you to perform the following during the creation of a view:

- Indicate the schema and name the view
- Formulate SQL statement, with optional assistance from SQLAssist
- Select check options
- Comment on the view

Create View statement



```

CONNECT TO TESTDB;
CREATE VIEW EMPSALARY
    AS SELECT EMPNO, EMPNAME, SALARY
    FROM PAYROLL, PERSONNEL
    WHERE EMPNO=EMPNUMB AND SALARY > 30000.00;

SELECT * FROM EMPSALARY

```

EMPNO	EMPNAME	SALARY
10	John Smith	1000000.00
20	Jane Johnson	300000.00
30	Robert Appleton	250000.00
...		

© Copyright IBM Corporation 2007

Figure 5-10. Create View statement

CF238.3

Notes:

A view is an alternate representation of data from one or more tables. It can include some or all of the columns contained in the tables on which it is defined.

To create a view, you must be connected to a database — either implicitly or explicitly — and the base tables or views upon which the view is based must previously exist.

Views can be created using the SQL statement CREATE VIEW.

You must have SYSADM, DBADM, CONTROL, or SELECT privilege on each base table to create a view. **Privileges on the base tables granted to groups are not checked to determine authorization to create a view.** However, if the base table has SELECT privilege given to PUBLIC, a view could be created. In addition, you must have the IMPLICIT_SCHEMA privilege or the CREATEIN privilege on the schema used.

Views may be used to exclude users from seeing certain data: rows or columns. The WHERE clause used in the CREATE VIEW statement determines which rows may be viewed by the user. The columns listed in the AS SELECT clause determine which columns may be viewed by the user.

Views can also be used to increase the access rights to data for a special user group.

Views may be used to improve performance. If a *difficult* SQL statement is to be used by users, it may be advantageous to create a view that is coded to utilize an index, or to ensure that a join is correctly coded.

Data for a view is not separately stored. The data is stored in the base tables.

When an object is dropped, views can become **inoperative** if they are dependent on that object. To recover an inoperative view, determine the SQL statement that was initially used to create the view. This information can be obtained from the SYSCAT.VIEWS.TEXT column. Recreate the view by using the CREATE VIEW statement with the same view name. Use the GRANT statement to regrant all privileges that were previously granted on the view. If you do not want to recover an inoperative view, you can explicitly drop it with the DROP VIEW statement.

An inoperative view only has entries in the SYSCAT.TABLES and SYSCAT.VIEWS catalog views. All entries in the SYSCAT.VIEWDEP, SYSCAT.TABAUTH, and SYSCAT.COLUMNS catalog views are removed.

```
CREATE VIEW view-name (column-name { ,column-name }) AS fullselect  
{ WITH [ CASCDED | LOCAL ] CHECK OPTION }
```

WITH CHECK OPTION specifies the constraint that every row that is inserted or updated through the view must conform to the definition of the view. WITH CHECK OPTION must not be specified if the view is read-only. If WITH CHECK OPTION is specified for an updatable view that does not allow inserts, then the constraint applies to update only. If WITH CHECK OPTION is omitted, the definition of the view is not used in the checking of any insert or update operations that use the view. Some checking might still occur during insert or update operations if the view is directly or indirectly dependent on another view that includes WITH CHECK OPTION.

CASCDED causes the constraints of all dependent views to also be applied.

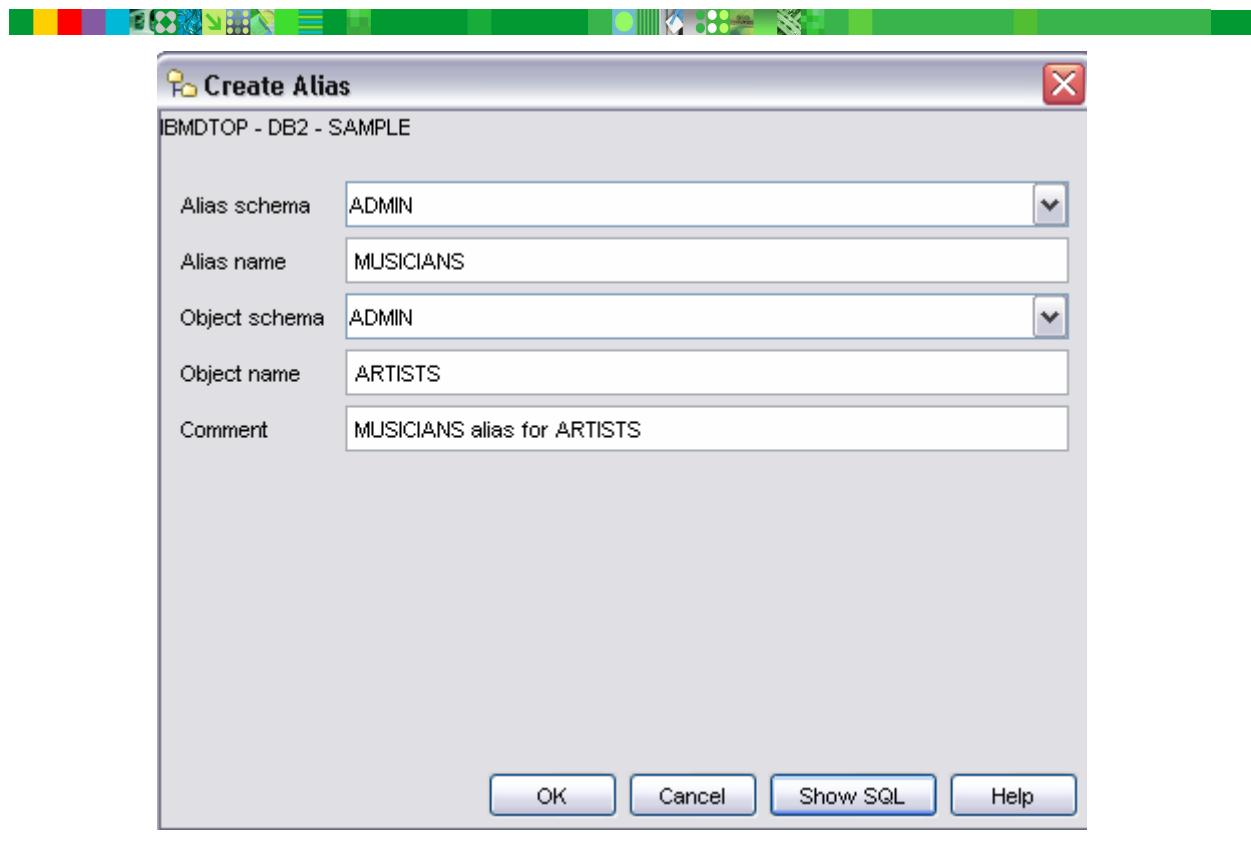
LOCAL causes the constraints of only this view to be applied.

A view may be defined on a view.

A read-only view cannot be the object of an INSERT, UPDATE, or DELETE statement.

For more information, refer to the *SQL Reference* manual.

Create Alias



© Copyright IBM Corporation 2007

Figure 5-11. Create Alias

CF238.3

Notes:

To access the Create Alias GUI from the Control Center:

- Select **Aliases** (right-click the DB node *Aliases*)
- Select **Create**

The Create Alias GUI allows you to perform the following during the creation of a alias:

- Select alias schema
- Provide alias name
- Identify schema and name of object (table, view, alias) on which the alias will be created
- Comment on the alias

Create Alias statement

- Cannot be the same as an existing table, view, or alias
- Can only refer to a table in the same database

```
CONNECT TO SAMPLE;
CREATE ALIAS ADMIN.MUSICIANS FOR ADMIN.ARTISTS;
COMMENT ON Alias ADMIN.MUSICIANS IS 'MUSICIANS
alias for ARTISTS';
CONNECT RESET;
```

© Copyright IBM Corporation 2007

Figure 5-12. Create Alias statement

CF238.3

Notes:

An alias is an indirect method of referencing a table or a view, so that an SQL statement can be independent of the qualified name of that table or view. Only the alias definition must be changed if the table or the view name changes.

An alias can be created on another alias.

An alias can be used in a view or trigger definition and in any SQL statements, except for constraint definitions, in which an existing table or view name can be referenced.

An alias can be defined for a table, view, or alias that does not exist. However, the object must exist when an SQL statement containing the alias is compiled.

When an alias, or the object to which an alias refers, is dropped, all packages dependent on the alias are marked invalid, and all views and triggers dependent on the alias are marked inoperative.

CREATE SYNONYM is an alternative for CREATE ALIAS. In DB2 for Linux, UNIX, and Windows, synonyms and aliases have exactly the same characteristics.

Overview of Declared Temporary Tables

```

CREATE USER TEMPORARY TABLESPACE
"USR_TEMP_TS"
PAGESIZE 4 K MANAGED BY AUTOMATIC STORAGE
BUFFERPOOL IBMDEFAULTBP ;

DECLARE GLOBAL TEMPORARY TABLE T1
LIKE REAL_T1
ON COMMIT DELETE ROWS
NOT LOGGED
IN USR_TEMP_TS;

INSERT INTO SESSION.T1
SELECT * FROM REAL_T1 WHERE DEPTNO=:mydept;

```

- /* do other work on T1 */
- /* when connection ends, table is automatically dropped */

© Copyright IBM Corporation 2007

Figure 5-13. Overview of Declared Temporary Tables

CF238.3

Notes:

A declared temporary table is a temporary table that is only accessible to SQL statements that are issued by the application which created the temporary table. A declared temporary table does not persist beyond the duration of the connection of the application to the database.

Use declared temporary tables to potentially improve the performance of your applications. When you create a declared temporary table, DB2 does not insert an entry into the system catalog tables, and, therefore, your server does not suffer from catalog contention issues. In comparison to regular tables, DB2 does not lock declared temporary tables or their rows. If your current application creates tables to process large amounts of data and drops those tables once the application has finished manipulating that data, consider using declared temporary tables instead of regular tables.

To use a declared temporary table, perform the following steps:

- Step 1: Ensure that a USER TEMPORARY TABLESPACE exists. If a USER TEMPORARY TABLESPACE does not exist, issue a CREATE USER TEMPORARY TABLESPACE statement. A USER TEMPORARY TABLESPACE stores declared

temporary tables. No user temporary table spaces exist when a database is created. At least one user temporary table space should be created with appropriate USE privileges to allow definition of declared temporary tables.

```
create user temporary tablespace usr_temp_ts managed by system  
using ('c:\usrspc')
```

- Step 2: Issue a DECLARE GLOBAL TEMPORARY TABLE statement in your application.

The definition on the graphic creates a user temporary table called T1. The user temporary table is defined with columns that have exactly the same name and description as the columns of the REAL_T1. The implicit definition only includes the column name, datatype, nullability characteristic, and column default value attributes. All other column attributes including unique constraints, foreign key constraints, triggers, and indexes are not defined. You can also specify column definitions as when creating a persistent table. See the *SQL Reference* for more information.

If you specify a schema for your temporary table, it must be SESSION.

Create Index

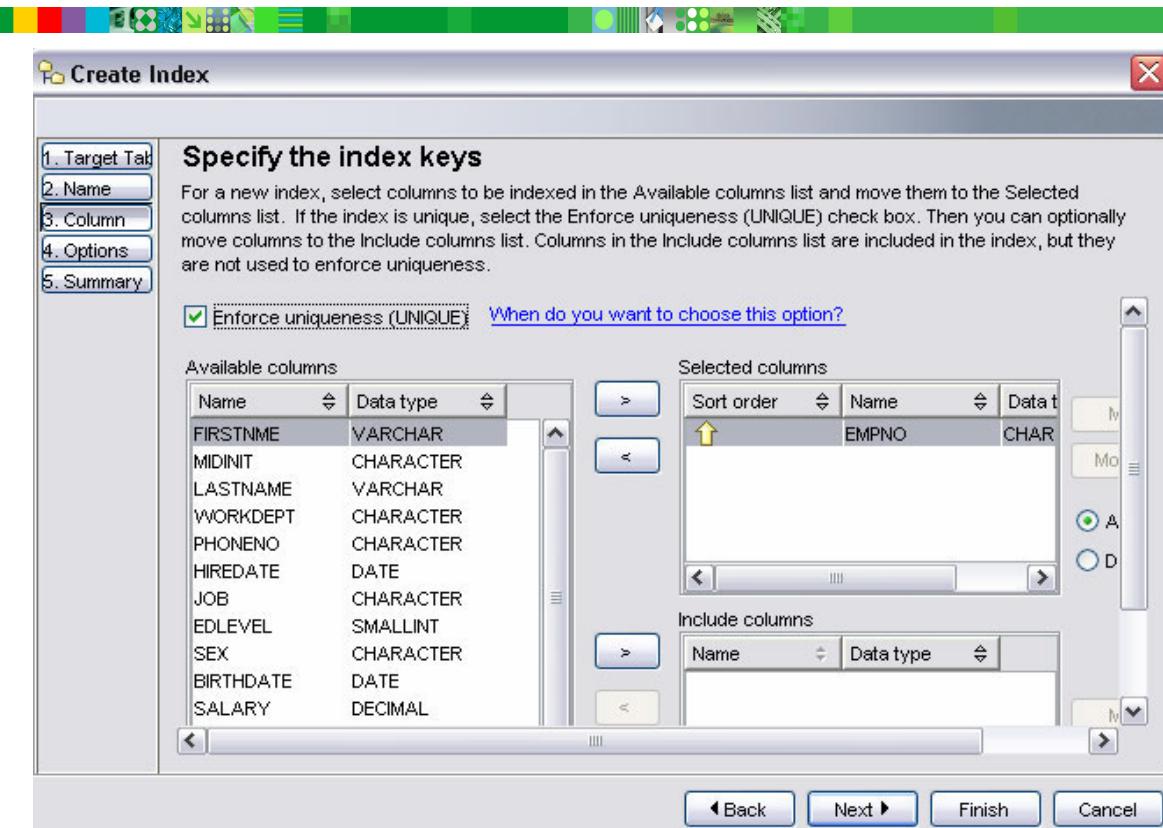


Figure 5-14. Create Index

CF238.3

Notes:

To access the Create Index GUI from the Control Center:

- Select **Indexes** (right-click the DB node *Indexes*)
 - Select **Create Index**

The Create Indexes GUI allows you to perform the following during creation of an index:

- Select index and table schemas
- Provide index name
- Identify table for which the index is to be created
- Select columns in the index; the sequence of selection determines the order of columns in the composite key
- Define ascending or descending order
- Define if the index is unique, clustered, or if it allows reverse scans
- Define the percentage of free space for index pages
- Define the minimum amount of used space prior to online index leaf page defragmentation
- Comment on the index

Create Index statement

```
create unique index dba1.empno on
  dba1.employee (empno asc)
  pctfree 10
  minpctused 10
  allow reverse scans
  page split symmetric
  collect sampled detailed statistics ;

create unique index itemno on albums (itemno)

create index item on stock (itemno) cluster

create unique index empidx on employee (empno)
  include (lastname, firstname)
```

© Copyright IBM Corporation 2007

Figure 5-15. Create Index statement

CF238.3

Notes:

Indexes may allow more efficient access to rows in a table. Unique indexes ensure uniqueness of the index key. An index key is a column or an ordered collection of columns on which an index is defined.

For an index, column-name identifies a column that is to be part of the index key. For an index specification, column-name is the name by which the federated server references a column of a data source table. Each column-name must be an unqualified name that identifies a column of the table. Up to 64 columns can be specified. If table-name is a typed table, up to 63 columns can be specified. If table-name is a subtable, at least one column-name must be introduced in the subtable; that is, not inherited from a supertable (SQLSTATE 428DS). No column-name can be repeated (SQLSTATE 42711). The sum of the stored lengths of the specified columns must not be greater than the index key length limit for the page size. For key length limits, see *SQL limits*. Note that this length limit can be reduced even more by system overhead, which varies according to the data type of the column and whether or not the column is nullable.

To create an index, you must be connected to a database — either implicitly or explicitly — and the base table upon which the index is based must previously exist.

Indexes can be created using the SQL statement CREATE INDEX.

You can use the Design Advisor to recommend indexes by identifying the SQL statements to be executed and the frequency of their use (that is, workload). The Design Advisor helps to improve performance of your database by suggesting and creating an optimal set of indexes.

You must have SYSADM, DBADM, CONTROL, or INDEX privilege on the table to create an index. You must also have either the IMPLICIT_SCHEMA privilege or the CREATEIN privilege on the schema used.

Indexes may be used to improve performance. An index can be used to search a table for matching data versus doing a table scan to search for matching data.

Indexes may be bidirectional (ALLOW REVERSE SCANS) and can support forward and backward scans. This is the default condition when creating an index. These indexes will facilitate MIN and MAX functions, allow fetch of the previous key, eliminate the need for the optimizer to create a temporary table for a reverse scan, and eliminate the need for a redundant reverse order index.

Data for an index is stored separately. The index columns are stored in the index with a pointer to the base table's row.

An index may be based on one or more columns in the table. An index may be stored in ASCending or DESCending order. Ascending order is the default.

The UNIQUE option prevents the table from containing two or more rows with the same value of the index key. The constraint is enforced when rows of the table are updated or new rows are inserted. The constraint is also checked during the execution of the CREATE INDEX statement. If the table already contains rows with duplicate key values, the index is not created. When UNIQUE is used, null values are treated as any other values. The column may contain no more than one null value.

By creating a unique index with include columns, you can improve the performance of data retrieval by increasing the use of index-only accesses.

A clustering index indicates that data records should be clustered on pages based on the sequence of the index, and maintains that clustering as much as possible over the course of insert activity. Clustering increases the efficiency of data retrieval when it involves accessing sequential value ranges for the clustering index.

If the named table already contains data, CREATE INDEX creates the index entries for it. If the table does not yet contain data, CREATE INDEX creates a description of the index; the index entries are created when data is inserted into the table.

```
CREATE [UNIQUE] INDEX index-name ON table-name
    (column-name [ASC|DESC] [, column-name])
    [ INCLUDE (column-name [, column-name]) ]
    [ CLUSTER ] [ PCTFREE integer ] [ MINPCTUSED integer ]
```

```
[ DISALLOW REVERSE SCANS | ALLOW REVERSE SCANS ]  
[ COLLECT [ [ SAMPLED ] DETAILED ] STATISTICS ]
```

The **PCTFREE** can be used to control the amount of free space left on an index page when an index is created. The default is 10% and the valid range of values is 0-90. By setting the value to a higher number, index page splits can be minimized. By setting the value to a lower number, fewer pages could be required to store the index for a read-only table.

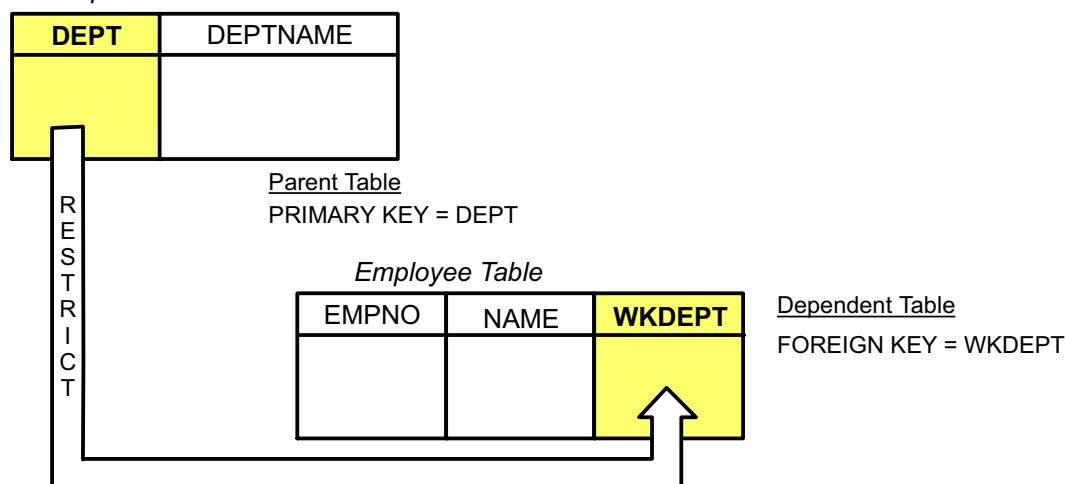
MINPCTUSED indicates whether indexes are defragmented online and the threshold for the minimum percent of space used on an index leaf page. If, after a key is deleted from an index leaf page, the percentage of space used on the page is at or below the percentage specified for MINPCTUSED, an attempt is made to merge the remaining keys on this page with those of a neighboring page.

PAGE SPLIT specifies an index split behavior. The default is SYMMETRIC, which specifies that pages are to be split roughly in the middle.

COLLECT STATISTICS specifies that index statistics are to be collected during index creation.

For more information, refer to the *SQL Reference* manual.

Overview of Referential Integrity



- Place constraints between tables
- Constraints specified with Create and Alter table statements
- Database services enforce constraints: inserts, updates, deletes
- Removes burden of constraint checking from application programs

© Copyright IBM Corporation 2007

Figure 5-16. Overview of Referential Integrity

CF238.3

Notes:

Referential Integrity (RI) lets you define required relationships between and within tables. The database manager maintains these relationships, which are expressed as *referential constraints*, and requires that all values of a given attribute or column of a table also exist in some other table or column. Since DB2 enforces RI, benefits include greater application development productivity (no code is required in the application to enforce RI) and data integrity and consistency, since DB2 consistently enforces the RI requirements in all environments.

In the event of deletion of a row in the parent table, constraints can be specified that the dependent rows are either completely deleted along with the parent data, or that only the dependent data is removed from the rows leaving a null value in the otherwise intact rows, or that the parent row cannot be deleted since dependent rows exist.

Primary Key is a unique, non-null key. There can be none or one primary key per table. The primary key can comprise one to 16 columns, maximum length is 1024 bytes, and all of the columns in the primary key must be defined as NOT NULL. When a primary key is specified, Database Services automatically creates a **unique index** in ascending order on

that key if one does NOT exist. The unique index name, if DB2 automatically creates it for you, will be SYSIBM.SQLyyyymmddhhmmssx (15-character timestamp). If a constraint name is specified, the index name is table_schema.constraint_name. The UNIQUERULE column in the SYSCAT.INDEXES catalog view will be **P**, representing a primary index.

A **Unique Constraint** is a rule that specifies that values of a key must be kept unique within the table. The columns specified in a unique constraint must be defined as NOT NULL. A unique index is used by DB2 to ensure uniqueness. A table can have any number of unique constraints, but only one on any given set of columns.

A **Parent Key** is a unique constraint or primary key that is referenced by a foreign key.

Foreign Key is a non-null value that matches the value of an associated parent key in the same table or a different table. The value is null if any nullable part of the key is null. A table can have none, one, or many foreign keys. Since more than one foreign key can be associated with a table, a **constraint name** may be defined by the user when the foreign key is defined. If no name is defined, Database Services assigns one using the format SQLnnnn where nnnn is a unique sequence (within the object table) of five numeric digits. The parent table with its parent key must exist before the foreign key is specified.

A **Parent Table** has a parent key that is referenced by one or more foreign keys.

A **Dependent Table** is a table with at least one foreign key.

A **Descendent Table** contains a foreign key that can be traced back to a parent key of a table. A table is a **descendent** if it is a dependent table or if it is a descendent of a dependent.

A **Dependent Row** is a row of a dependent table that has a non-null foreign key value that matches a parent key value of its parent table.

A **Parent Row** is a row of a parent table whose parent key value matches at least one foreign key value in a dependent table. When insert, update, and delete operations are performed on a parent table:

1. A row can only be inserted into a parent table if the value of the parent key is non-null and unique.
2. The parent key value of a parent row cannot be updated. The old row would need to be deleted and a new row inserted.
3. When a row is deleted from a parent table, a check is performed to determine if there are any dependent rows in the dependent table with matching foreign key values. If any dependent rows are found, the action taken is prescribed by the **Delete Rule** specified in the referential constraint.

Delete Rules

- a. **Restrict or No Action** — A row of a parent table cannot be deleted if one or more rows exist in the dependent tables which have a foreign key value matching the parent key value.

- b. **Cascade** — If a row in a parent table is deleted, then all rows in the dependent tables with a foreign key matching the parent key value of this row will also be deleted.
- c. **Set Null** — If a row in a parent table is deleted, then all rows in the dependent tables with a foreign key matching the parent key value of this row will have their foreign key value changed to null. All columns of the foreign key which are nullable will be set to null. SET NULL can only be specified if at least one of the columns comprising the foreign key is nullable.

RI — Add a foreign key

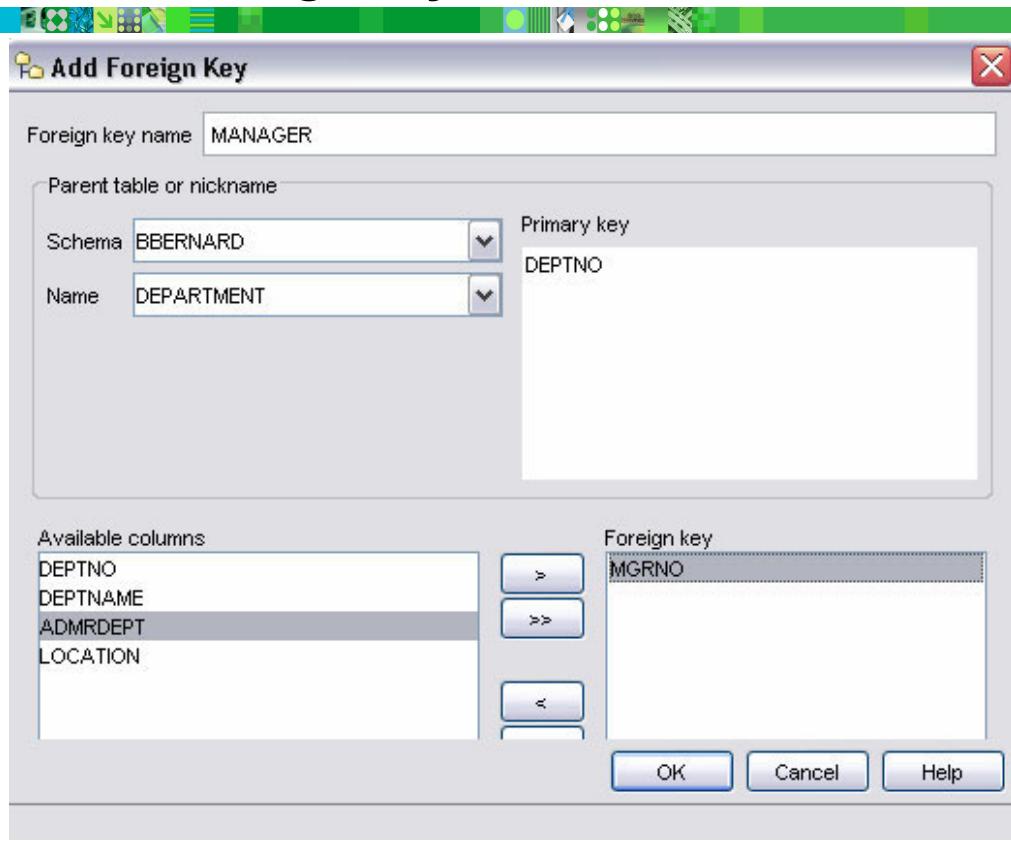


Figure 5-17. RI — Add a foreign key

CF238.3

Notes:

To create a referential constraint during table creation:

- Right-click and select **Tables**
 - Select **Create**
 - Select Primary Keys where you can choose what to create (Primary, Unique, Foreign, or Partitioning key)
- Primary key is set on the parent table and foreign key is set on the dependent table.

A referential constraint can also be created between tables through the alter table GUI.

The other possibility is to highlight the table you want to change, right-click and select **ALTER**. There you will find a *KEYS* tab where you have the same choices as above.

Referential Integrity — Create Table statement

 CREATE TABLE DEPARTMENT

```
(DEPT CHAR (3) NOT NULL,
DEPTNAME CHAR (20) NOT NULL,
CONSTRAINT DEPT_NAME
UNIQUE (DEPTNAME),
PRIMARY KEY (DEPT))
```

IN DMS99

CREATE TABLE EMPLOYEE

```
(EMPNO CHAR (6) NOT NULL,
NAME CHAR (30),
WKDEPT CHAR (3) NOT NULL,
CONSTRAINT DEPT
FOREIGN KEY
(WKDEPT)
REFERENCES DEPARTMENT
ON DELETE RESTRICT)
```

IN SMS99 ...

© Copyright IBM Corporation 2007

Figure 5-18. Referential Integrity — Create Table statement

CF238.3

Notes:

A unique constraint on any column or set of columns will not allow duplicate values in those columns. In the example above, a unique index called DEPT_NAME will automatically be created in the same schema to enforce the unique constraint.

To create a table and include referential constraints in its definition, SYSADM, DBADM, or CREATETAB privilege on the database and REFERENCES or CONTROL privilege on the parent table are required. The CREATE TABLE statement allows specification of referential constraints at table creation time.

To create referential constraints on an existing table, at least SYSADM, DBADM, or CONTROL or ALTER privilege on the dependent table plus REFERENCES privilege on the parent table are required. The ALTER TABLE statement allows specification of referential constraints on existing tables. Data already in the table must meet the constraints, or the constraint cannot be created. The parent key must be defined on the parent table before a related foreign key may be defined.

A referential constraint is implicitly dropped:

- When the parent or dependent table of the relationship is dropped
- When the definition of the parent key of the parent table is dropped.

A second table could be defined that allows for look-up of department functions by department name, with a referential constraint on the DEPARTMENT.DEPTNAME as its parent key:

```
CREATE TABLE dept_function
  keyword      CHAR(25) NOT NULL,
  name         CHAR(20) NOT NULL,
  CONSTRAINT dn FOREIGN KEY (name)
    REFERENCES department (deptname)
  ON DELETE CASCADE
) IN sms01
```

Unique Key considerations

- Multiple keys in one table can be foreign key targets

```
CREATE TABLE PAY.EMPTAB
  (EMPNO SMALLINT NOT NULL PRIMARY KEY,
   NAME CHAR(20),
   DRIVERLIC CHAR(17) NOT NULL,
   CONSTRAINT DRIV_UNIQ UNIQUE(DRIVERLIC)
  ) IN TBSP1
```

Unique indexes PAY.DRIV_UNIQ and SYSIBM.yymmddhhmmssxxx
created Columns must be NOT NULL

- Deferral of unique checking until end of statement processing
 - `UPDATE EMPTAB SET EMPNO=EMPNO + 1`

© Copyright IBM Corporation 2007

Figure 5-19. Unique Key considerations

CF238.3

Notes:

A unique constraint is the rule that the values of a key are valid only if they are unique within the table. Unique constraints are optional and can be defined in the CREATE TABLE or ALTER TABLE statement using the PRIMARY KEY clause or the UNIQUE clause. The columns specified as a unique constraint must be defined as NOT NULL. A unique index is used by the database manager to enforce the uniqueness of the key during changes to the columns of the unique constraint.

A table can have an arbitrary number of unique constraints, with at most one unique constraint defined as a primary key. A table cannot have more than one unique constraint on the same set of columns.

A unique constraint that is referenced by the foreign key of a referential constraint is called a parent key. When a unique constraint is defined in a CREATE TABLE statement, a unique index is automatically created by the database manager and designated as a primary or unique system-required index. When a unique constraint is defined in an ALTER TABLE statement and an index exists on the same columns, that index is designated as unique and system-required. If such an index does not exist, the unique index is

automatically created by the database manager and designated as a primary or unique system-required index. CONSTRAINT constraint-name identifies the constraint. If this clause is omitted, an 18-character identifier, unique within the identifiers of the existing constraints defined on the table, is generated by the system. When used with a PRIMARY KEY or UNIQUE constraint, the constraint-name will be used as the name of an index that is created to support the constraint if a unique index does not exist. If the CONSTRAINT constraint-name clause is not used, the system-generated index will be named SYSIBM.SQLyyymddhhmmssx.

PRIMARY KEY provides a shorthand method of defining a primary key composed of a single column. Thus, if PRIMARY KEY is specified in the definition of column C, the effect is the same as if the PRIMARY KEY(C) clause is specified as a separate clause.

UNIQUE provides a shorthand method of defining a unique key composed of a single column. Thus, if UNIQUE is specified in the definition of column C, the effect is the same as if the UNIQUE(C) clause is specified as a separate clause.

The unique constraint clause defines a unique or primary key constraint.

CONSTRAINT constraint-name names the primary key or unique constraint.

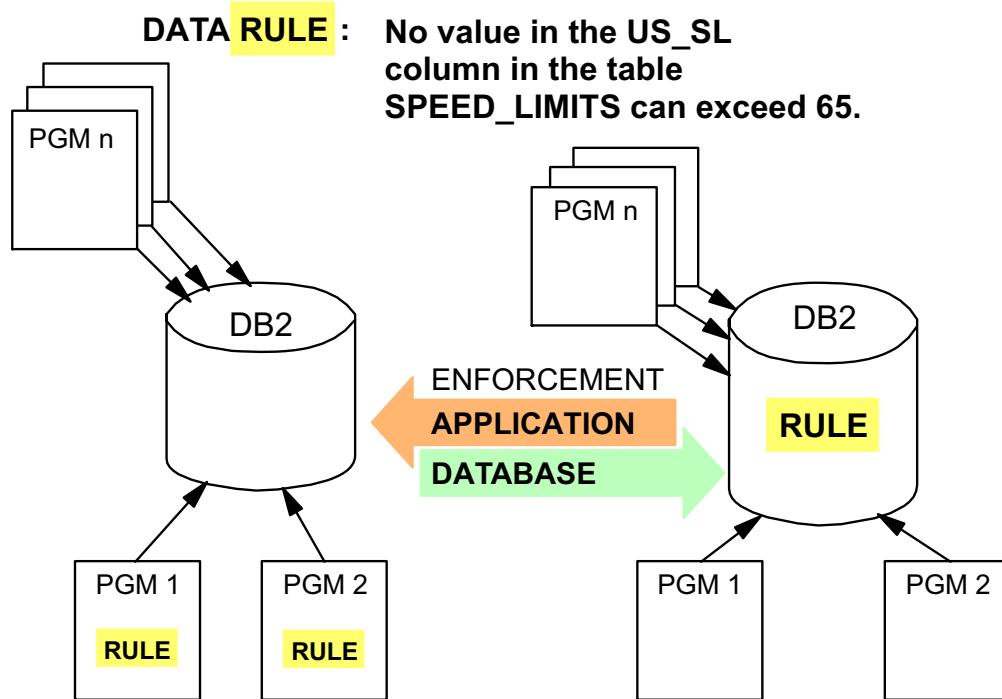
UNIQUE(column-name, ...) defines a unique key composed of the identified columns. The identified columns must be defined as NOT NULL.

If a column or set of columns is defined as being unique, and an update statement is issued which updates a set of rows, DB2 will not check for uniqueness until all of the rows have been updated (within the single statement). This will allow a statement like

```
UPDATE EMPTAB SET EMPNO = EMPNO + 1
```

to execute successfully, even if there are consecutive values currently found in the EMPTAB table (which would cause duplicate values within the table for a short duration, during the processing of the UPDATE statement).

Overview of Check Constraints: The need



© Copyright IBM Corporation 2007

Figure 5-20. Overview of Check Constraints: The need

CF238.3

Notes:

Referential Integrity is available to enforce relationships between one table and another through parent key and foreign key definitions. However, many data rules are specific to the column values in a single table. Check constraints can be used to define such rules to the DB2 database manager so that rules are enforced by the database manager.

Constraints will be checked during INSERT and UPDATE operations.

The source of the INSERT or UPDATE operation is not relevant for the defined constraint to be checked. Any application, whether installation-defined, vendor-provided, or ad hoc, will have INSERT and UPDATE statements validated such that the values in constrained columns meet the definition of the constraints.

Installations requiring such data checking should consider using the database manager support. The constraint is defined once, relieving the application programmer from coding such checks or invoking other code to do such checks.

Check Constraints — definition

```
CREATE TABLE SPEED_LIMITS
(ROUTE_NUM  SMALLINT,
 CANADA_SL  INTEGER NOT NULL,
 US_SL       INTEGER NOT NULL
CHECK (US_SL <=65) )
```

```
ALTER TABLE SPEED_LIMITS
ADD
CONSTRAINT SPEED65
CHECK (US_SL <=65)
```

© Copyright IBM Corporation 2007

Figure 5-21. Check Constraints — definition

CF238.3

Notes:

Column constraints can be defined using the SQL statements CREATE TABLE or ALTER TABLE. The constraint name cannot be the same as any other constraint specified within that statement, and must be unique within the table.

If the ALTER TABLE statement is used, existing data is checked against the new constraint before the ALTER statement succeeds. If any rows exist that would violate the constraint, the ALTER TABLE statement fails.

To add constraints to a large table, it is more efficient to put the table into the check pending state, add the constraints, and then check the table for a consolidated list of violation rows. Use the SET INTEGRITY statement to explicitly set the check pending state. If the table is a parent table, check pending is implicitly set for all dependent and descendent tables.

When a table check constraint is added, packages that insert or update the table may be marked as invalid. More details on packages and implicit rebind will be presented in the Application Alternatives unit.

The definition of the constraint allows basic WHERE clause constructs to be used:

- Basic comparisons (>, <, =, >=, and so on)
- BETWEEN
- LIKE
- IN
- Deterministic UDFs

Values can only be inserted or updated in the column if the result of the constraint test resolves to True.

The definition of the constraint does not support:

- subqueries
- column functions
- functions that are not deterministic
- functions defined to have an external action
- user defined functions defined with either CONTAINS SQL or READS SQL DATA
- host variables or parameter markers
- special registers (such as CURRENT DATE)
- references to generated columns other than the identity column

The constraint can be explicitly named when it is defined. If it is not named, DB2 will create a name.

The ALTER TABLE statement can also be used to DROP constraints. For example:

```
ALTER TABLE SPEED_LIMITS DROP CONSTRAINT SPEED65
```

or

```
ALTER TABLE SPEED_LIMITS DROP CHECK SPEED65
```

Implementation of Check Constraints

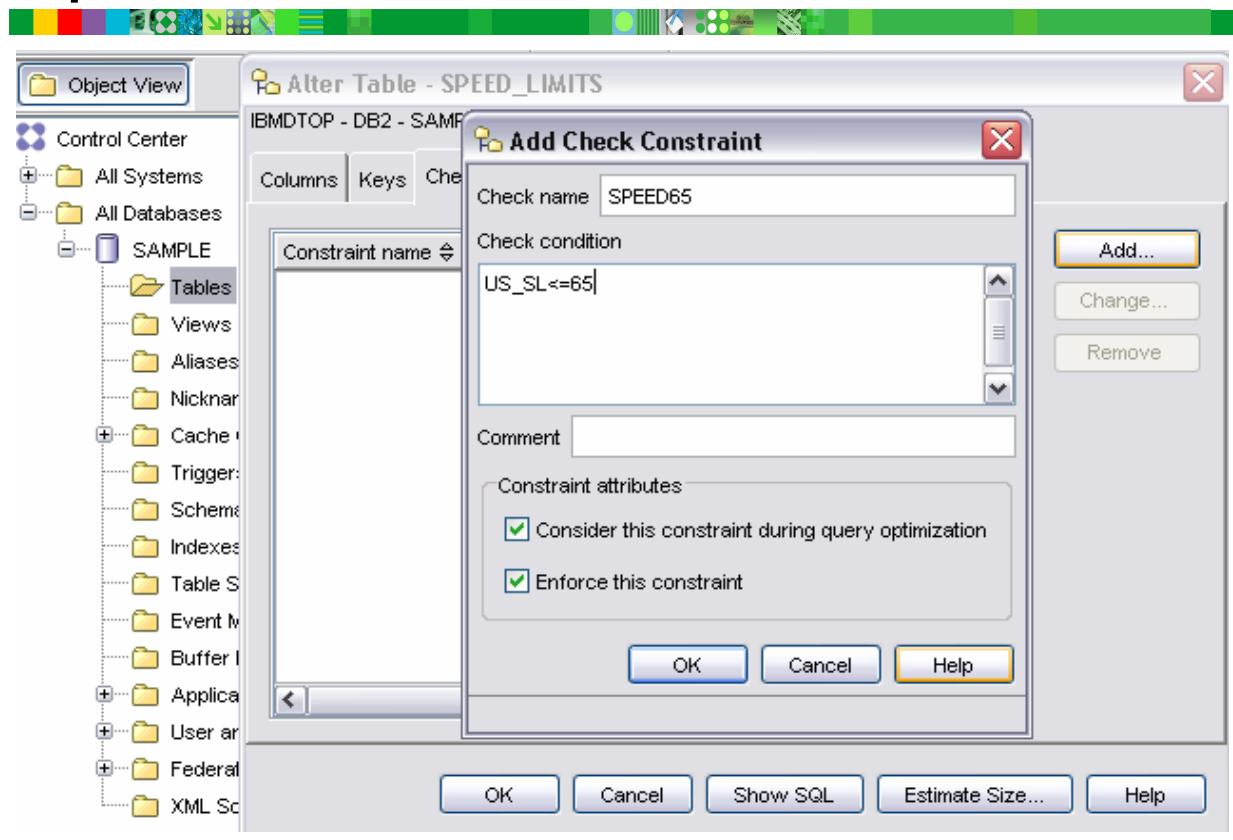


Figure 5-22. Implementation of Check Constraints

CF238.3

Notes:

To create a check constraint:

- Left-click and select **Tables**

Left-click and select the table you plan to work with from the *Contents* pane

- Select **Alter**

From the Alter Table GUI, you can add Check Constraints to the selected table.

Check constraints can also be specified when the table is initially being created.

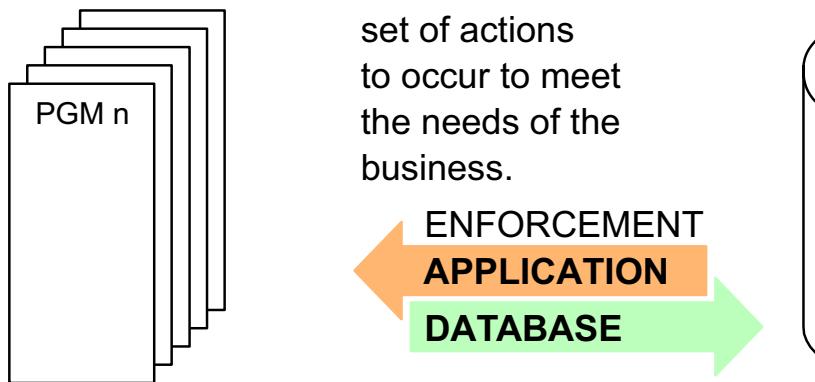
Overview of triggers: The need



BUSINESS RULES:

**INSERT, UPDATE,
and DELETE**

actions against a table may require another action or set of actions to occur to meet the needs of the business.



© Copyright IBM Corporation 2007

Figure 5-23. Overview of triggers: The need

CF238.3

Notes:

Triggers can be defined to DB2 so that the database manager automatically enforces business rules that should follow an INSERT, UPDATE, or DELETE operation against the specified table. The benefits of DB2 enforcement of such rules include eliminating the need for redundant coding, global enforcement of the defined rules, and ease of maintenance if the business rules change.

Business rules that may be supported by triggers include:

- Validation of input
- Automatic generation of a value for an inserted row
- Reading from other tables for cross-referencing purposes
- Writing to other tables for audit-trail purposes
- Maintaining business rules
- Providing support for user alerts

Implementation of triggers (1 of 2)

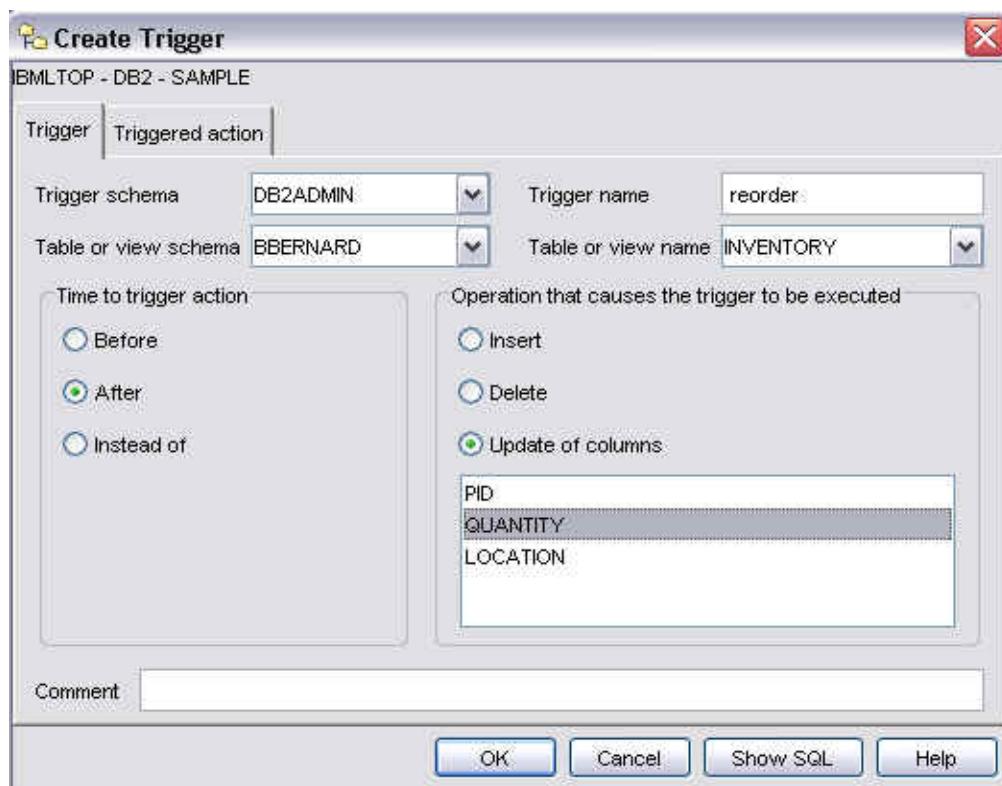


Figure 5-24. Implementation of triggers (1 of 2)

CF238.3

Notes:

To access the Create Trigger GUI from the Control Center:

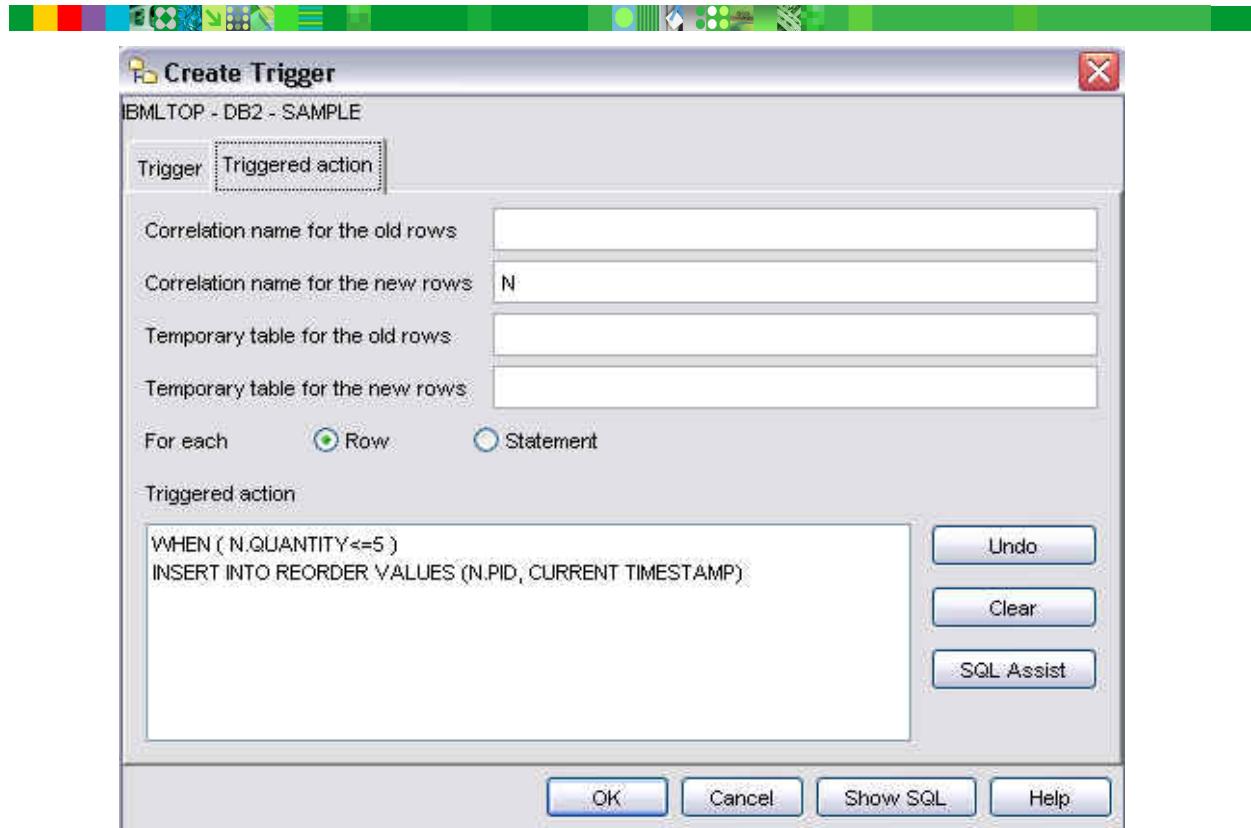
- Right-click and select **Triggers**
 - Select **Create**

The Create Triggers GUI has the following two tabs:

- Trigger
- Triggered action

The Create Trigger GUI allows you to create and define triggers on tables in the database.

Implementation of triggers (2 of 2)



© Copyright IBM Corporation 2007

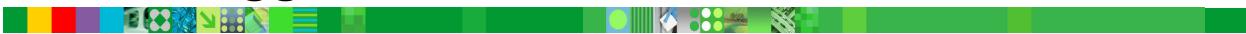
Figure 5-25. Implementation of triggers (2 of 2)

CF238.3

Notes:

The Triggered action allows you to define what should occur when the trigger is invoked.

Create Trigger statement



```
create trigger reorder
    after update
        of qty on stock
    referencing new as n
    for each row mode db2sql
    when (n.qty <=5)
insert into reorder values (n.itemno, current
timestamp)
```

© Copyright IBM Corporation 2007

Figure 5-26. Create Trigger statement

CF238.3

Notes:

A trigger defines a set of actions that are executed or *triggered* by an UPDATE, INSERT, or DELETE on a specified table. The triggered action is specified by an SQL procedure statement. The SQL procedure statement can contain a dynamic compound statement or any of the SQL control statements listed in “Compound SQL (Dynamic)” in the *SQL Reference* manual.

If the trigger is a BEFORE trigger, then an SQL procedure statement can also include a fullselect, or a SET variable statement. If the trigger is an AFTER trigger, then an SQL Procedure statement can also include one of the following:

- an INSERT statement (not using nicknames)
- a searched UPDATE statement (not using nicknames)
- a searched DELETE statement (not using nicknames)
- a SET variable statement
- a fullselect (a common table expression may precede a fullselect)

In the visual above, the CREATE TRIGGER statement has the following options:

CREATE TRIGGER reorder (trigger-name) specifies the name of the trigger. After creation you can also find this name in the system catalog tables, like SYSCAT.TRIGGERS and SYSCAT.TRIGDEP.

AFTER indicates that the triggered action happens after the changes are made to the table.

NO CASCADE BEFORE (not on visual) specifies that the triggered action is to be applied before any changes caused by the update are applied to the table. Also, this specifies that the action of the trigger should not cause other triggers to be activated.

INSERT/UPDATE/DELETE are the SQL operations which will cause the trigger to potentially fire.

OF qty (column-name) controls the granularity of the triggering, to specify a specific column rather than the whole table.

ON stock (table) specifies the base table to which the trigger applies.

REFERENCING NEW AS n (correlation name) is used to provide correlation names which are used as transaction variables. A specification of a correlation name for the variables prior to the triggering SQL operation can be given with **REFERENCING OLD AS o**. A temporary table name can be given for the set of affected rows prior to the triggering SQL operation by using **REFERENCING OLD_TABLE AS identifier**; the set of affected rows, as modified by the triggering SQL operation, and by any SET statement in a BEFORE trigger that has already executed, can be identified by **REFERENCING NEW_TABLE AS identifier**.

FOR EACH ROW/STATEMENT indicates that the trigger should be applied once for each row of the subject table that is affected by the triggering SQL operation (ROW), or that the triggered action is to be applied only once for the whole statement (statement).

MODE DB2SQL is a required parameter to specify the mode of the trigger. Only DB2SQL is currently supported.

WHEN (n.qty <= 5) (search-condition) specifies a condition that is true, false, or unknown. The search-condition provides the capability to determine whether or not a certain triggered action should be executed.

INSERT INTO reorder VALUES (n.itemno, CURRENT TIMESTAMP) (SQL-procedure-statement) is the SQL that should be executed if the search condition is true. The SQL procedure statement can include SQL control statements like FOR, IF, ITERATE, LEAVE, or WHILE to implement procedural logic, as well as statements such as GET DIAGNOSTICS, SIGNAL, and SET variable to interact with DB2. One of the column values here is the value of the current timestamp.

Triggers can also have multiple conditions which may be checked for via the CASE statement.

Adding a trigger to a table that already has rows in it will not cause any triggered actions to be activated. This may be of concern if the trigger is intended to enforce complex data integrity rules.

The creation of a trigger will cause packages to be invalidated if the package contains a statement that is now a potential triggering action.

Triggers can be removed with the DROP TRIGGER statement. Any packages that had a dependency on the trigger will be marked invalid.

Triggers are limited to a depth of 16 cascaded triggers. If a trigger at level 17 is activated, a negative SQLCODE (-724) will be returned and the triggering statement will be rolled back.

There is overhead for each additional trigger.

In order to create a trigger, you must have SYSADM or DBADM authority; or have ALTER privilege on the table; or have ALTERIN privilege on the schema and one of either IMPLICIT_SCHEMA privilege on the database if the schema of the trigger does not exist, or CREATEIN privilege on the schema, if the schema name of the trigger refers to an existing schema.

Large objects: The need

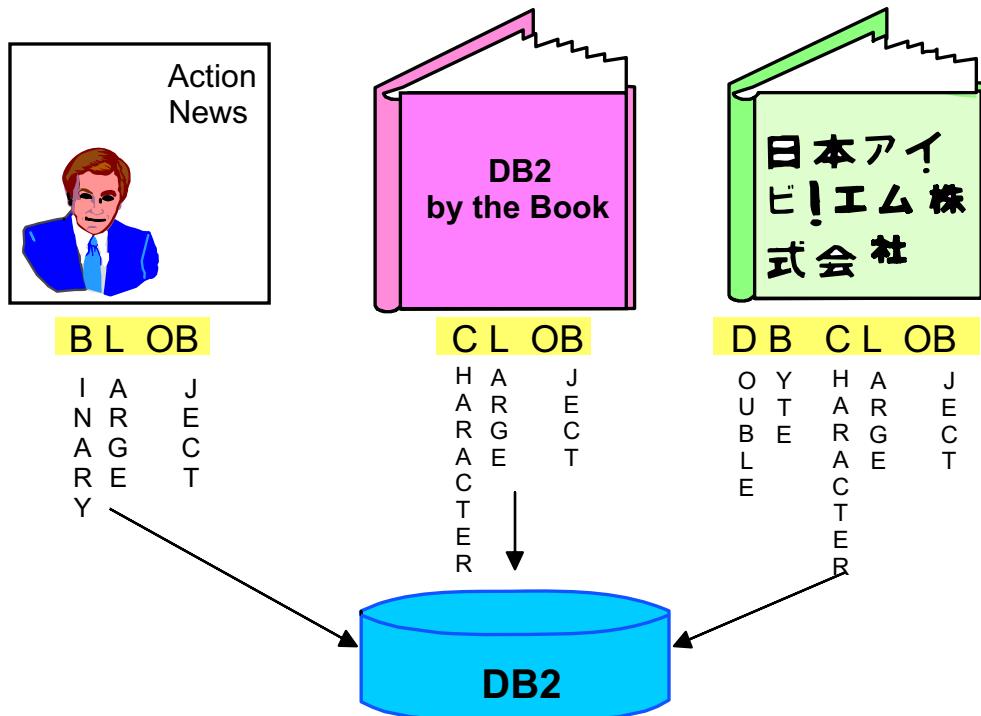


Figure 5-27. Large objects: The need

CF238.3

Notes:

As advancements in hardware allow greater volumes and types of data to be stored, installations require such data types to be handled as part of a relational system.

- LOB values can be very large, and the transfer of these values from database to client can be time consuming
- Because application programs typically process LOB values one piece at a time, rather than as a whole, applications can reference a LOB value by using a large object locator
- A large object locator, or LOB locator, is a host variable whose value represents a single LOB value on the database server
- An application program can select a LOB value into a LOB locator
- Then, using the LOB locator, the application program can request database operations on the LOB value by supplying the locator value as input
- The resulting output (data assigned to a client host variable) would typically be a small subset of the input LOB value

Large Objects, collectively known as **LOBs**, include binary large objects (**BLOBs**), character large objects (**CLOBs**), and double-byte character large objects (**DBCLOBs**).

- BLOBs can be used to store images and voice data. For example, an application could store the headline stories from a news broadcast that could then be selectively reviewed.
- CLOBs can be used any time large amounts of character (usually text) data are to be stored. For example, an application could store the text of popular books or online help text. (Can contain single- or double-byte characters).
- DBCLOBs are much like CLOBs, except that they store data in the double-byte character set. For example, the online help text mentioned for the CLOB data might need to be stored in Kanji. (Can contain double- or four-byte characters).

DB2 supports multiple LOB columns. The data for LOB columns is stored directly in the database.

Each LOB column can hold various sizes of data, with a maximum size of 2 GB.

On the CREATE TABLE, you can define whether or not changes to LOB data are to be LOGGED. Since LOB data by definition implies large data volumes, it may be advantageous not to log changes to LOB data. LOB data greater than 1 GB cannot be logged, and LOB data greater than 10 MB probably should not be logged.

Indicating COMPACT on the definition of a LOB specifies that the values in the LOB column should take up minimal disk space (any extra disk pages in the last group used by the LOB value should be freed), rather than leave any leftover space at the end of the LOB storage area that might facilitate subsequent append operations. Note that storing data in this way may cause a performance penalty in any append (length-increasing) operations on the column.

An example of DDL to define LOB columns is shown:

```
CREATE TABLE newsclip (
    title VARCHAR(100) NOT NULL,
    storydate DATE,
    footage BLOB(4500000) NOT LOGGED,
    audio BLOB(1000000) NOT LOGGED,
    text CLOB(50000) COMPACT
)
```

BLOB (integer [K|M|G]) specifies the maximum length in bytes for a binary large object. The length may be in the range of 1 byte to 2 GB.

If [integer K] is specified, the maximum length is 1024 (1 KB) times integer.

If [integer M] is specified, the maximum length is 1,048,576 (1 MB) times integer.

If [integer G] is specified, the maximum length is 1,073,741,824 bytes (1 GB) times integer.

CLOB (integer [K|M|G]) is for a character large object string. The meaning of integer K|M|G is the same as for BLOB.

DBCLOB (`integer [K|M|G]`) is for a double-byte-character large object string. The meaning of `integer K|M|G` is similar to that for a BLOB. The differences are that the value specifies the number of double-byte characters and that the maximum size is 1,073,741,823 (1 GB) double-byte characters.

5.3 Creating tables with XML columns

What is XML? (1 of 2)



- XML = eXtensible Markup Language
- XML is *self-describing data*
- Here is some data from an ordinary delimited flat file:

```
47; John Doe; 58; Peter Pan; Database systems; 29; SQL;
      relational
```

- What does this data mean?

```
<book>
  <authors>
    <author id="47">John Doe</author>
    <author id="58">Peter Pan</author>
  </authors>
  <title>Database systems</title>
  <price>29</price>
  <keywords>
    <keyword>SQL</keyword>
    <keyword>relational</keyword>
  </keywords>
</book>
```

XML: Describes data
HTML: Describes display

© Copyright IBM Corporation 2007

Figure 5-28. What is XML? (1 of 2)

CF238.3

Notes:

The example of data above – taken from a flat file – shows a line of raw data. We see the numbers 47, 58, 29 and the words John Doe, Peter Pan – but we don't know what each of these pieces of data means; the context of the data is external to the data itself.

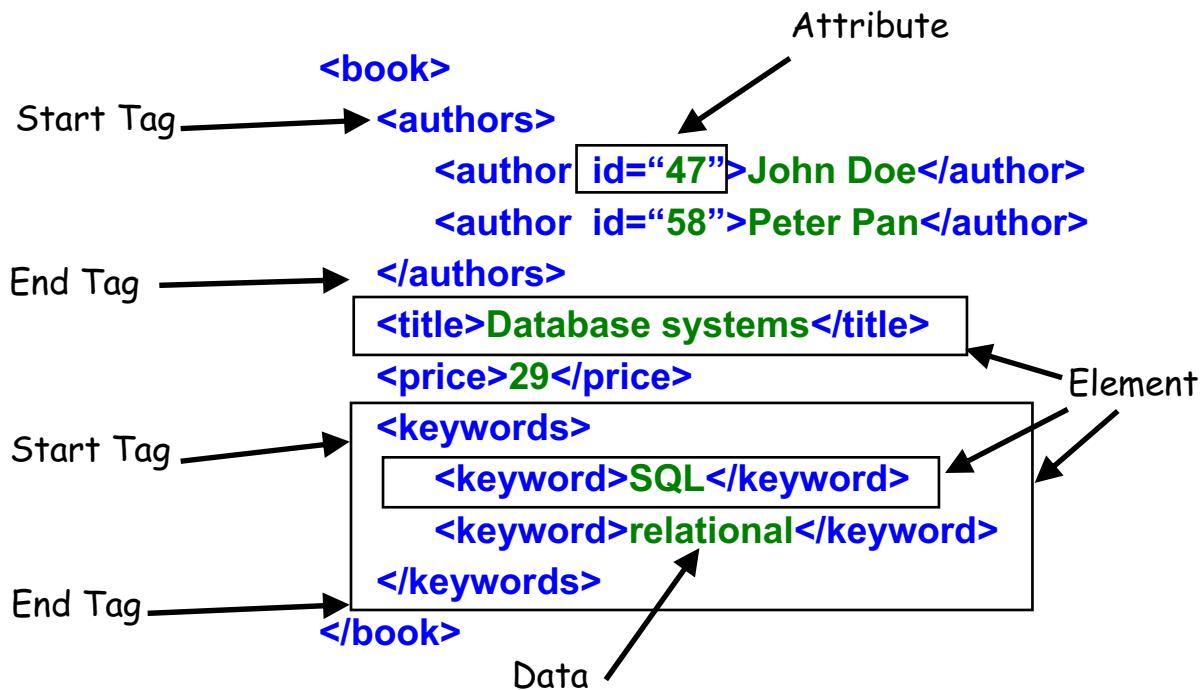
The XML markup shown above is in blue and the actual data is in red. By viewing both, we know what this data means.

XML is eXtensible Markup Language. The markup shows the tags used to describe the data. XML is a self-describing kind of data.

The key difference between HTML and XML markup languages:

- XML describes the relationship of the data.
- HTML describes how to display the data in a browser.
- XML is extensible – this means you can make up your own markup and tags to describe the data. You can create your own tags if they are not tied to a certain schema.
- HTML has a fixed vocabulary of tags and can only use those tags.

What is XML? (2 of 2)



© Copyright IBM Corporation 2007

Figure 5-29. What is XML? (2 of 2)

CF238.3

Notes:

Elements and attributes make up an XML document. These elements always have a start and stop tag. Tags are always in angle brackets <> and the end tag is denoted by the slash /. Elements can be nested.

The root element is the outer most element for the document – as shown above, the **book** element is the root element.

Attributes can be specified for an XML element. As shown above, the **author** element has an **id** attribute assigned to it. The attributes are always defined in the start tag. In the code shown above, the value of the **id** attribute for the first author is **47** and assigned to the **author** element in the first set of angle brackets.

Note that the **authors** element (denoted by the **authors** tag) contains two nested elements named **author**.

The XML document tree

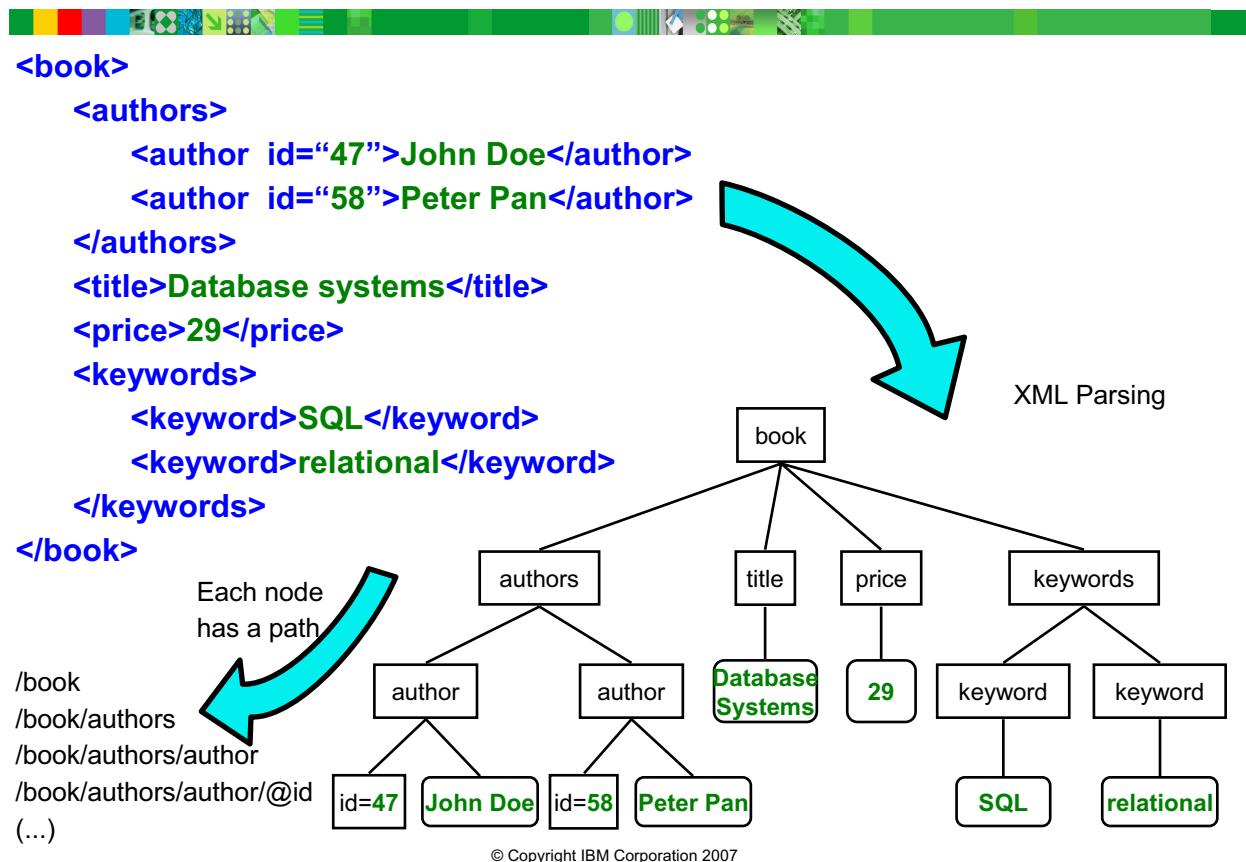


Figure 5-30. The XML document tree

CF238.3

Notes:

Every XML document can be viewed as a tree – a hierarchy of nodes. Every XML document can be seen as a higher key of nodes. It may be intuitive that this represents a hierarchy of nodes.

In the example above, **book** is the root node of the tree. The data elements are always the leaf nodes in the tree (such as the **title** element in the example above).

XML parsing typically produces this tree form. This is how XML can be manipulated by software — storing XML in trees in DB2 and not as text, which is more efficient and faster to manipulate. This is done by parsing the tree.

Serialization is reversing the parsing operation, which takes a tree and returns it to XML document format.

In the graphic shown above, each node would have the following paths:

- root node (book element)

/book

- authors node (authors element)
/book/authors
 - author node (author element)
/book/authors/author (two members)
/book/authors/author/@id (author attribute = id)
 - author text node (text element)
/book/authors/author/text (values are *John Doe, Peter Pan*)
- title node (title element)
/book/title
 - title text node (text element)
/book/title/text (value is *Database System*)
- price node (price element)
/book/price
 - price text node (text element)
/book/price/text (value is 29)
- keywords node (keywords element)
/book/keywords
 - keyword node (keyword element)
/book/keywords/keyword (two members)
 - keyword text node (text element)
/book/keywords/keyword/text (values are *SQL, relational*)

PureXML datastore



- PureXML's seamless integration of XML with relational data:
 - Speeds application development
 - Improves search performance with highly optimized XML indexes
 - Is flexible because both SQL and XQuery can be used to query XML data
 - Enables well-formed XML documents to be stored in their hierarchical form
- Stored XML data can be accessed and managed by leveraging DB2 functionality
 - Enables efficient search and retrieval of XML
- Note current limitation:
 - XML data can only be stored in single-partition databases defined with the UTF-8 code set

© Copyright IBM Corporation 2007

Figure 5-31. PureXML datastore

CF238.3

Notes:

The pure XML support in DB2 offers efficient and versatile capabilities for managing your XML data. DB2 stores and processes XML in its inherent hierarchical format, avoiding the performance and flexibility limitations that occur when XML is stored as text in CLOBs or mapped to relational tables. Unlike XML-only databases, DB2 9 also provides seamless integration of relational data and XML data within a single database, even within a single row of a table. This flexibility is reflected in the language support, which allows you to access relational data, XML data, or both at the same time. You can query your XML in any of the following four ways:

- Plain SQL (no XQuery involved)
- SQL/XML, that is, XQuery embedded in SQL
- XQuery as a stand-alone language (no SQL involved)
- XQuery with embedded SQL

The native XML data store enables well-formed XML documents to be stored in their hierarchical form within columns of a table. XML columns are defined with the XML data

type. By storing XML data in XML columns, the data is kept in its native hierarchical form, rather than stored as text or mapped to a different data model.

Because the native XML data store is fully integrated into the DB2 database system, the stored XML data can be accessed and managed by leveraging DB2 functionality.

The storage of XML data in its native hierarchical form enables efficient search and retrieval of XML. XQuery, SQL, or a combination of both can be used to query XML data. SQL functions that return XML data or take XML arguments (referred to as SQL/XML functions) also enable XML data to be constructed or published from values retrieved from the database.

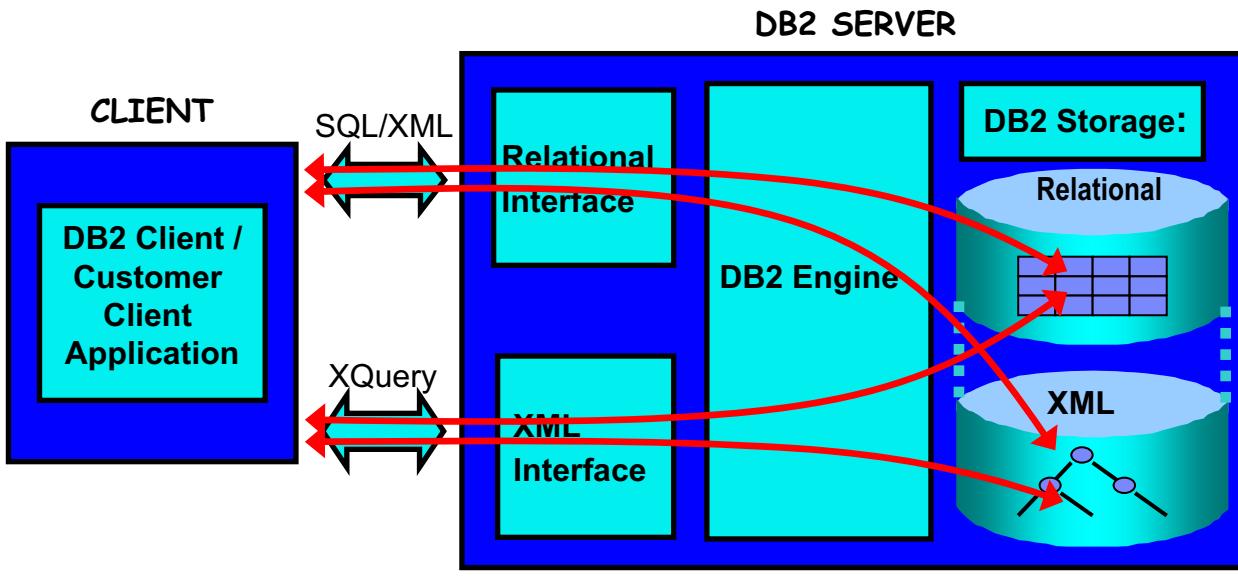
XML data can only be stored in single-partition databases defined with the UTF-8 code set.

Note that using XML features prevents future use of the Database Partitioning Feature available with DB2 Enterprise Server Edition for Linux, UNIX, and Windows.

Use of the new XML data type and related native XML data store support is available as a separate feature of DB2 9. You must acquire the same licensing terms and conditions as the underlying DB2 data server.

Integration of XML and relational capabilities

- Applications combine XML and relational data
- Native XML data type (server and client side)
- XML capabilities in all DB2 components



© Copyright IBM Corporation 2007

Figure 5-32. Integration of XML and relational capabilities

CF238.3

Notes:

The goal:— integration of XML and relational capabilities so that applications can combine XML and relational data.

A new DB2 datatype: XML type

- The XML type is visible on the server and the client.
- There is deep integration as the figure above shows. This is not an add-on, extender, or wrapper.
- The DB2 infrastructure is reused wherever it made sense, with new system code where needed.
- DB2 supports a new query language: XQuery, with a new parser (side by side of SQL parser).
- XML data is natively stored: hierarchical, optimized for parent/child traversal.

XML permeates the engine. You can use SQL or XQuery to query a mix of relational and XML data stored in DB2.

The same complier handles both XML and relational data, so SQL and XQuery queries can be mixed.

Because XML is not just another numeric type, but a significant change, special efforts are necessary in almost all system components, tools and utilities to handle XML.

Xquery:

Xquery is an emerging standard for access to XML data. It is designed, from the ground up, to handle semi-structured data.

Two ways to query XML data



- Using XQuery as the primary language

```
XQUERY db2-fn:xmlcolumn ("CUSTOMER.INFO")
      Returns all Documents in column INFO
```

- Using SQL as the primary language

```
Select Pid from PRODUCT xmlp
WHERE XMLExists('declare default element namespace
  "http://posample.org";$p/product/description[price
  < 100]'
  passing xmlp.description as "p");
```

© Copyright IBM Corporation 2007

Figure 5-33. Two ways to query XML data

CF238.3

Notes:

You may use a new query language – XQuery – to retrieve XML documents (and parts of documents) from the database.

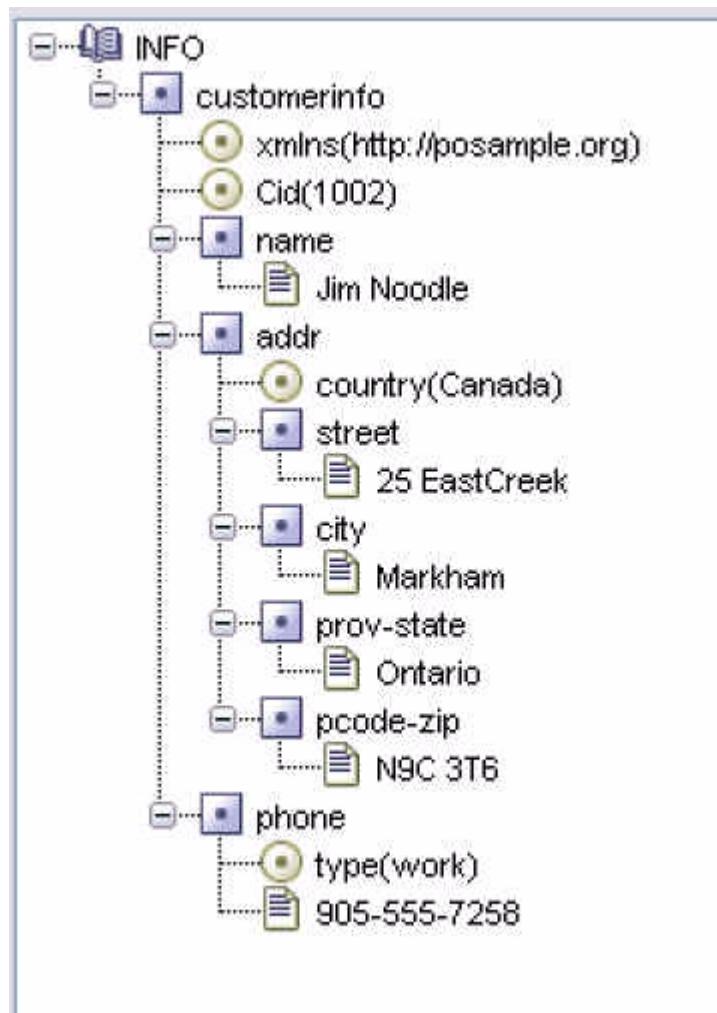
Example1– Xquery:

XQUERY db2-fn:xmlcolumn ("CUSTOMER.INFO")

Results - serial view:

```
...
<customerinfo xmlns="http://posample.org" Cid="1002">
<name>
  Jim Noodle
</name>
<addr country="Canada">
<street>
  25 EastCreek
</street>
```

```
<city>
    Markham
</city>
<prov-state>
    Ontario
</prov-state>
<pcode-zip>
    N9C 3T6
</pcode-zip>
</addr>
<phone type="work">
    905-555-7258
</phone>
</customerinfo>
...
Tree view:
```



You may also use SQL to retrieve XML documents and relational database data, and combine those type of information in one query (SQL/XML).

Example2 – SQL/XML:

```
Select Pid from PRODUCT xmlp
  WHERE XMLExists('declare default element namespace
"http://posample.org";$p/product/description[price < 100]' passing
xmlp.description as "p");
```

Results - serial view:

PID
100-100-01
100-101-01
100-103-01
100-201-01

4 record(s) selected.

For more information



- To learn more about DB2 and XML, including the use of xQuery and SQL/XML:

- Course CG13 — 4 days
Query and Manage XML Data with DB2 9
- Course CG10 — 2 days
Query XML data with DB2 9
- Documentation:
DB2 XML Guide (SC10-4254-00)
DB2 XQuery Reference (SC18-9796-00)



© Copyright IBM Corporation 2007

Figure 5-34. For more information

CF238.3

Notes:

Query XML data with DB2 9 — CG10

Duration: 2 days

Course description:

This course is designed to introduce the students to non-traditional ways of using DB2 to work with native XML data.

The students query XML data stored in DB2 tables and create XML documents from traditional DB2 data. They search XML documents stored in DB2 tables and access XML documents as though the documents were relational tables. This course can be taken by end users or other personnel who only need to access XML data stored in DB2 tables.

Query and Manage XML Data with DB2 9 — CG13

Duration: 4 days

Course description:

This course is designed to introduce the students to non-traditional ways of using DB2 to work with native XML data. It is also designed to enable database administrators to manage the DB2 — native XML environment.

The students query XML data stored in DB2 tables and create XML documents from traditional DB2 data. They search XML documents stored in DB2 tables and access XML documents as though the documents were relational tables. This content is addressed in the first two days of class and can be taken by users or other personnel who only need to access XML data stored in DB2 tables.

In the last two days of class, the students store native XML data in DB2 and create XML indexes. They analyze plan information for queries and take steps for improving query execution. The course also teaches how to manage the XML schemas and the techniques available for decomposing the XML documents.

DB2 SQL Workshop — CF12

Duration: 2 days

Course description:

This course provides an introduction to the SQL language and applies to the entire DB2 Family. This course is appropriate for customers working in all DB2 environments, that is, z/OS, VM/VSE, iSeries, Linux, UNIX, and Windows.

RELATIONAL DATABASE DESIGN — CF18

Duration: 4 days

Course description:

This course presents a methodology for modeling and designing relational databases.

DB2 SQL Workshop for Experienced Users — CF13

Duration: 2.5 days

Course description:

This course teaches you how to make use of advanced SQL techniques to access DB2 databases in different environments. This course is appropriate for customers working in all DB2 environments, that is z/OS, OS/390, VM/VSE, iSeries, Linux, UNIX, and Windows.

Building the Data Warehouse — DW11

Duration: 4.5 days

Course description:

This course teaches students practical methods and techniques for designing and constructing a data warehouse. Students will learn how to apply the techniques in the context of an incremental data warehouse development process which is suitable for constructing a data warehouse from a departmental perspective. In addition, students will also learn how this process can further be extended for the construction of consistent,

corporate-wide data warehouses. As such, the course is suitable for students who wish to learn how to construct a data mart as well as a corporate-wide, integrated data warehouse.

During the course, students will apply the techniques they learn doing several exercises, which are part of a case study which is consistently developed throughout the course. This approach greatly enhances the practical value of the course, by allowing students to apply the acquired knowledge in practice, under the guidance of the instructor.

Unit summary

Having completed this unit, you should be able to:

- List DB2 object hierarchy and physical directories and files
- Learn to create the following objects:
 - Schema, Table, View, Alias, Index
- Explore the use of table partitioning
- Review the use of Temporary Tables
- Explore the use and implementation of Check Constraints, Referential Integrity and Triggers
- Exploring Large Objects: the need and usage
- Study an overview of XML as a native store

© Copyright IBM Corporation 2007

Figure 5-35. Unit summary

CF238.3

Notes:

Unit 6. Moving data

What this unit is about

This unit provides information on tools that can be used to manage table data, including the Export Utility, Import Utility, Load Utility, and the DB2MOVE utility.

The handling of set integrity pending conditions will also be addressed.

What you should be able to do

After completing this unit, you should be able to:

- Discuss the INSERT statement and recognize its limitations
- Explain the differences between IMPORT and LOAD
- Explain the EXPORT, IMPORT, and LOAD syntax
- Create and use Exception Tables and Dump-Files
- Distinguish and resolve Table States:
 - Load Pending and Set Integrity Pending
- Use the SET INTEGRITY command
- Discuss the db2move and db2look commands

How you will check your progress

Accountability:

- Machine exercises

References

Command Reference

Data Movement Utilities Guide and Reference

Replication Guide and Reference

Unit objectives



After completing this unit, you should be able to:

- Review the INSERT statement and recognize its limitations
- Review differences between IMPORT and LOAD
- Learn the EXPORT, IMPORT, and LOAD syntax
- Explore the use of Exception Tables and Dump-Files
- Distinguish the Table States:
 - Load pending, Set Integrity Pending
- Learn why and how to use the SET INTEGRITY command
- Explore use of the **db2move** and **db2look** commands

© Copyright IBM Corporation 2007

Figure 6-1. Unit objectives

CF238.3

Notes:

Review INSERT statement



- The SQL INSERT statement can insert one or more rows of data into tables, nicknames, or views
 - SQL overhead is imposed, such as obeying RI, or Check Constraints, or Uniqueness, or executing triggers.
 - As INSERTs occur, the activity is also stored in logs
- The SQL INSERT may not be the best or fastest method to load massive amounts of data into a database

© Copyright IBM Corporation 2007

Figure 6-2. Review INSERT statement

CF238.3

Notes:

The INSERT statement inserts rows into a table, nickname, or view, or the underlying tables, nicknames, or views of the specified fullselect. Inserting a row into a nickname inserts the row into the data source object to which the nickname refers.

Inserting a row into a view also inserts the row into the table on which the view is based, if no INSTEAD OF trigger is defined for the insert operation on this view. If such a trigger is defined, the trigger will be executed instead.

6.1 Export/Import utilities

Differences between IMPORT and LOAD



IMPORT	LOAD
<ul style="list-style-type: none"> ▪ Slow when moving large amounts of data ▪ Creation of table/index supported with IXF format ▪ Import into tables, views, aliases ▪ Option to ALLOW WRITE ACCESS ▪ All rows logged ▪ Triggers fired, constraints enforced 	<ul style="list-style-type: none"> ▪ Faster for large amounts of data ▪ Tables and indexes must exist ▪ Load into tables only ▪ Existing data can still be seen by read applications ▪ Minimal logging; can make copy ▪ Triggers not fired; unique constraints enforced, RI and check constraints via SET INTEGRITY

© Copyright IBM Corporation 2007

Figure 6-3. Differences between IMPORT and LOAD

CF238.3

Notes:

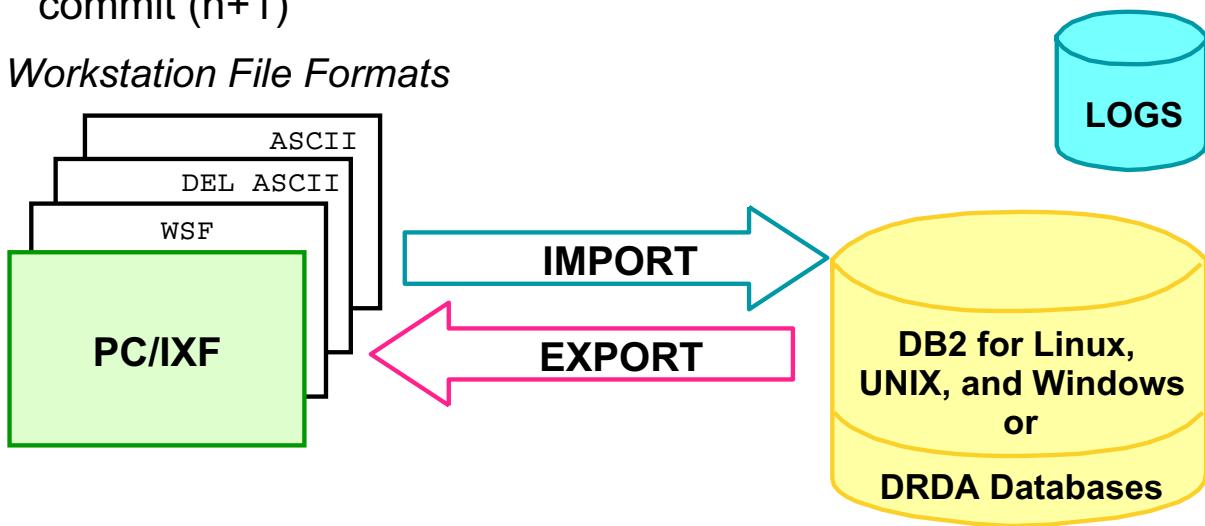
The import utility performs SQL INSERTs, so its capabilities are similar to an application program performing inserts. The load utility formats the pages and writes them directly into the database.

A more detailed version of this chart can be found in the *Data Movement Guide and Reference*.

IMPORT/EXPORT overview

- Must be connected prior to call
- IMPORT COMMITCOUNT n (AUTOMATIC) keeps log sizes manageable
- IMPORT RESTARTCOUNT n allows to restart from last commit (n+1)

Workstation File Formats



© Copyright IBM Corporation 2007

Figure 6-4. IMPORT/EXPORT overview

CF238.3

Notes:

The IMPORT utility may be used to insert data from an input file into a table, with the input file containing data from another database or spreadsheet program.

The EXPORT utility may be used to copy data from a table to an output file for use by another database or spreadsheet program. This file can be used to load tables, providing a convenient method of migrating data from one database to another.

The IMPORT and EXPORT utilities may be used to move data between databases which exist on DB2 Database platforms. These utilities use the database engine, so, for example, you could create a table during the execution of the IMPORT utility.

The IMPORT and EXPORT utilities may be used to move data between DB2 and DRDA host databases if DB2 Connect is installed.

Import and Export File Formats

- **DEL** — Delimited ASCII is commonly used for storing data that separates column values with a special delimiting character.

- **ASC** — Non-delimited ASCII may be used for importing data from other applications which create flat text files with aligned column data. **ASC may not be used with EXPORT.**
- **WSF** — Work sheet format is used for exchange with products such as Lotus 1-2-3 and Lotus Symphony.
- **IXF** — PC version of the Integrated Exchange Format. This is the preferred method for exchange within the database manager. Use PC/IXF to export data from a table so that it can be imported later into the same table or another table.

For IXF data file formats, the table does not need to exist before beginning the import. The table can automatically be created when the data is imported. For DEL, WSF, and ASC data file formats, the table including its column names and data types must be defined before the file can be imported.

For restrictions see: Using IMPORT to create an exported table.

During export, a UDT's base type will be stored in IXF files. If using the IXF file to create a new table during an import, the new table will have the base type for the column definitions instead of the UDT. The import utility supports importing values into a UDT column when the base data type supplied in the input file is castable to the type of the UDT.

For EXPORT and IMPORT, you can select LOB column types and have the data stored in separate files if the MODIFIED BY option LOBSINFILE is specified. The same path as the data file will be used.

XML FROM xml-path - Specifies one or more paths that contain the XML files.

XMLPARSE - Specifies how XML documents are parsed. If this option is not specified, the parsing behavior for XML documents will be determined by the value of the CURRENT XMLPARSE OPTION special register.

EXPORT command

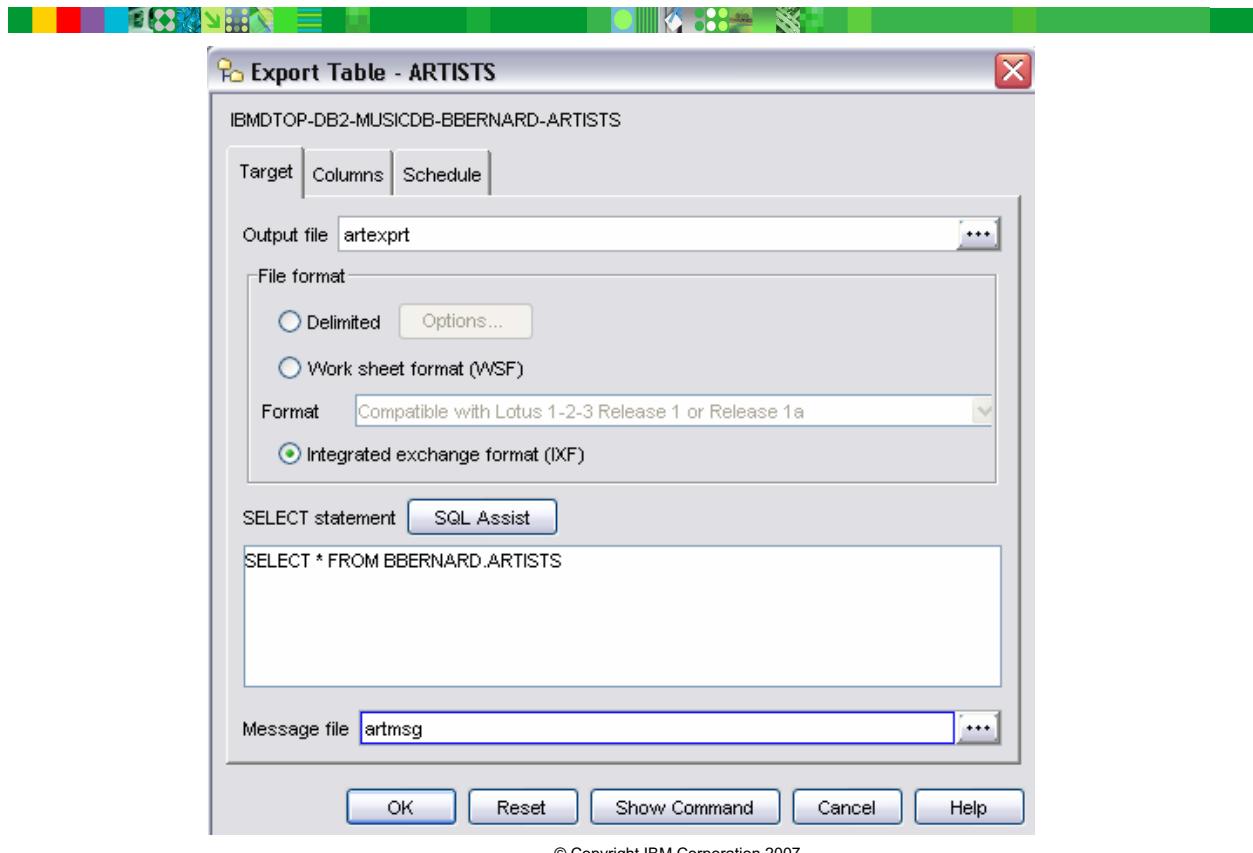


Figure 6-5. EXPORT command

CF238.3

Notes:

To export data through GUI:

- Left-click and select **Tables**

Right-click and select the table with which you plan to work from the *Contents* pane

- Select **Export**

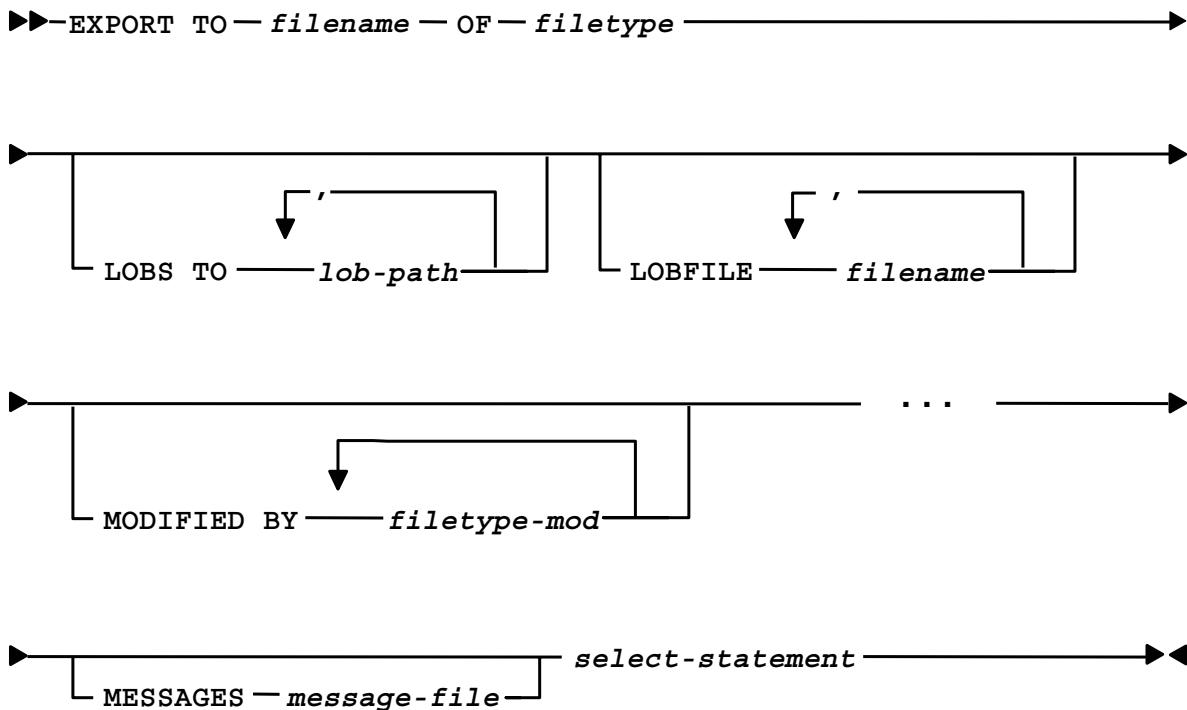
From the Export GUI, you can export data from a table to a file.

You will find the following tabs in the Export GUI.

- Target
- Columns
- Schedule

The Export GUI allows you to export data from tables to files, place large objects in separate files, define which columns should be exported, define the type of export, and name the message file. You can run the export now, or schedule it to run once or on a repeating schedule.

EXPORT command syntax (basic)



© Copyright IBM Corporation 2007

Figure 6-6. EXPORT command syntax (basic)

CF238.3

Notes:

The EXPORT command can be used to export data from a database to one of several external file formats. The user specifies the data to be exported by supplying an SQL SELECT statement.

File types that are supported include:

- **DEL (delimited ASCII format)** which is used by a variety of database manager and file manager programs.
- **WSF (work sheet format)** which is used by programs such as Lotus 1-2-3 and Lotus Symphony.
- **IXF (integrated exchange format, PC version)** in which most of the table attributes, as well as any existing indexes, are saved in the IXF file, except when columns are specified in the SELECT statement. With this format, the table can be recreated, while with the other file formats, the table must already exist before data can be imported into it.

There are many restrictions to the EXPORT command for table re-creation, such as you cannot export XML documents, and using SELECT * FROM table only.

The MODIFIED BY options allow you to specify different items depending on the file type being created. For example, for delimited output data, you can specify the character string delimiter and the column delimiter. See the *Data Movement Utilities Guide and Reference* or *Command Reference* for details.

Some EXPORT command parameters:

LOBS TO lob-path

Specifies one or more paths to directories in which the LOB files are to be stored. There will be at least one file per LOB path, and each file will contain at least one LOB. The maximum number of paths that can be specified is 999.

LOBFILE filename

Specifies one or more base file names for the LOB files. When name space is exhausted for the first name, the second name is used, and so on. The maximum number of file names that can be specified is 999.

MODIFIED BY filetype-mod

Specifies file type modifier options.

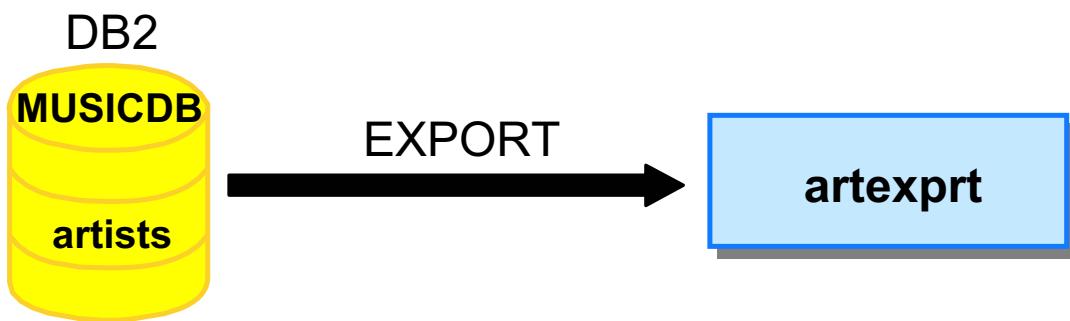
lobsinfile	lob-path specifies the path to the files containing LOB data.
xmlinsefiles	Each XQuery Data Model (QDM) instance is written to a separate file. By default, multiple values are concatenated together in the same file.
lobsinsefiles	Each LOB value is written to a separate file. By default, multiple values are concatenated together in the same file.
xmlnodedeclaration	QDM instances are written without an XML declaration tag. By default, QDM instances are exported with an XML declaration tag at the beginning that includes an encoding attribute.
xmlchar	QDM instances are written in the character codepage. Note that the character codepage is the value specified by the CODEPAGE file type modifier, or the application codepage if it is not specified. By default, QDM instances are written out in Unicode.
xmlgraphic	QDM instances are written in the graphic codepage. Note that the graphic codepage is the graphic component of the value specified by the CODEPAGE file type modifier, or the graphic component of the application codepage if it is not specified. By default, QDM instances are written in Unicode.

MESSAGES message-file

Specifies the destination for warning and error messages that occur during an export operation (the path must exist).

EXPORT command example

- Exports data from database tables to file
- Check message for error or warning messages



```
db2 connect to musicdb
db2 export to artexppt
  of ixf messages artmsg
  select artno, name
    from artists
```

© Copyright IBM Corporation 2007

Figure 6-7. EXPORT command example

CF238.3

Notes:

To export, you must have SYSADM, DBADM, or CONTROL or SELECT on each table or view that is referenced.

Any SELECT statement may be used to get information to be exported, but only SELECT * FROM one table exported to IXF results in a recreatable IXF file.

IMPORT command

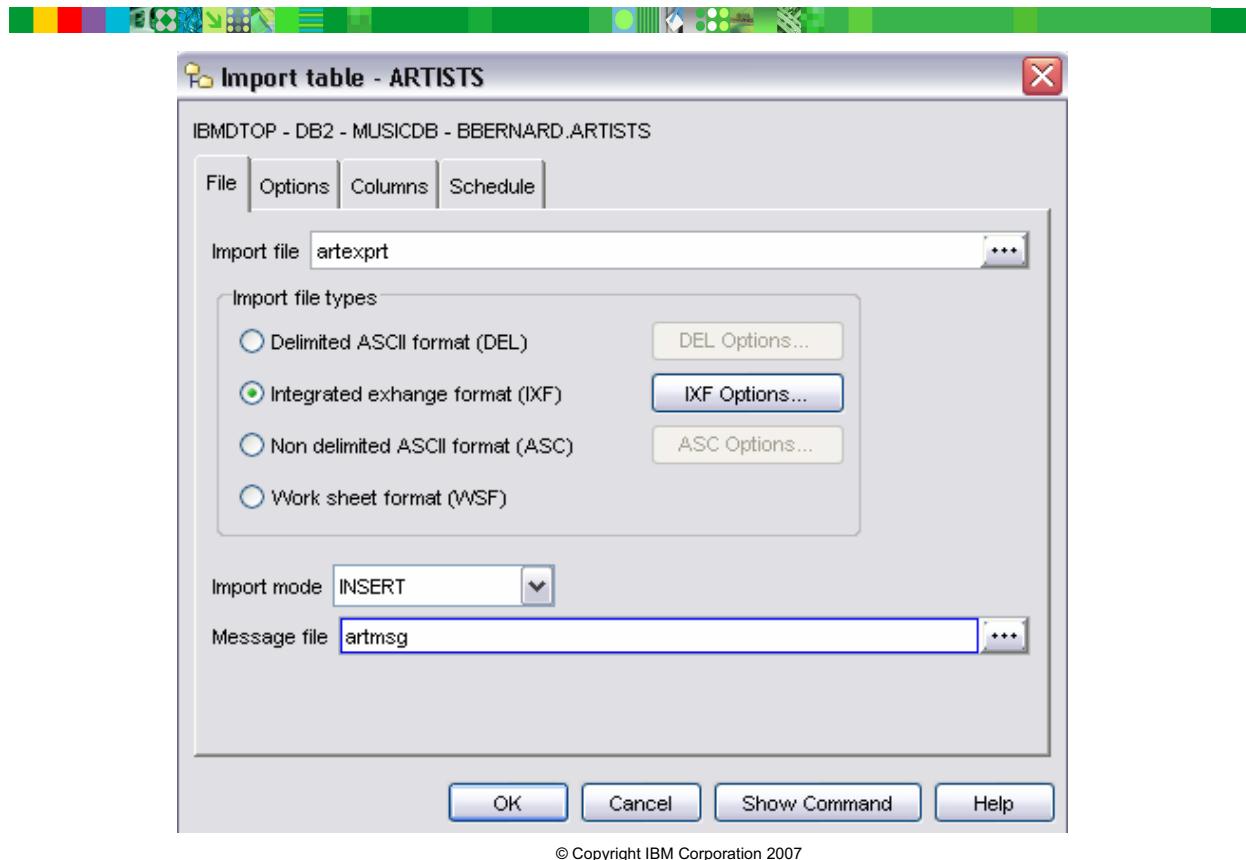


Figure 6-8. IMPORT command

CF238.3

Notes:

To Import data through GUI:

- Left-click and select **Tables**

Right-click and select the table with which you plan to work with from the *Contents* pane

- Select **Import**

From the Import GUI, you can import data from a file into a table.

You will find the following tabs in the Import GUI.

- File
- Columns
- Options
- Schedule

The Import GUI allows you to import data from files to tables, retrieve large objects from separate files, define which columns will be imported, define the method of import, and name the message file.

IMPORT command syntax (basic)

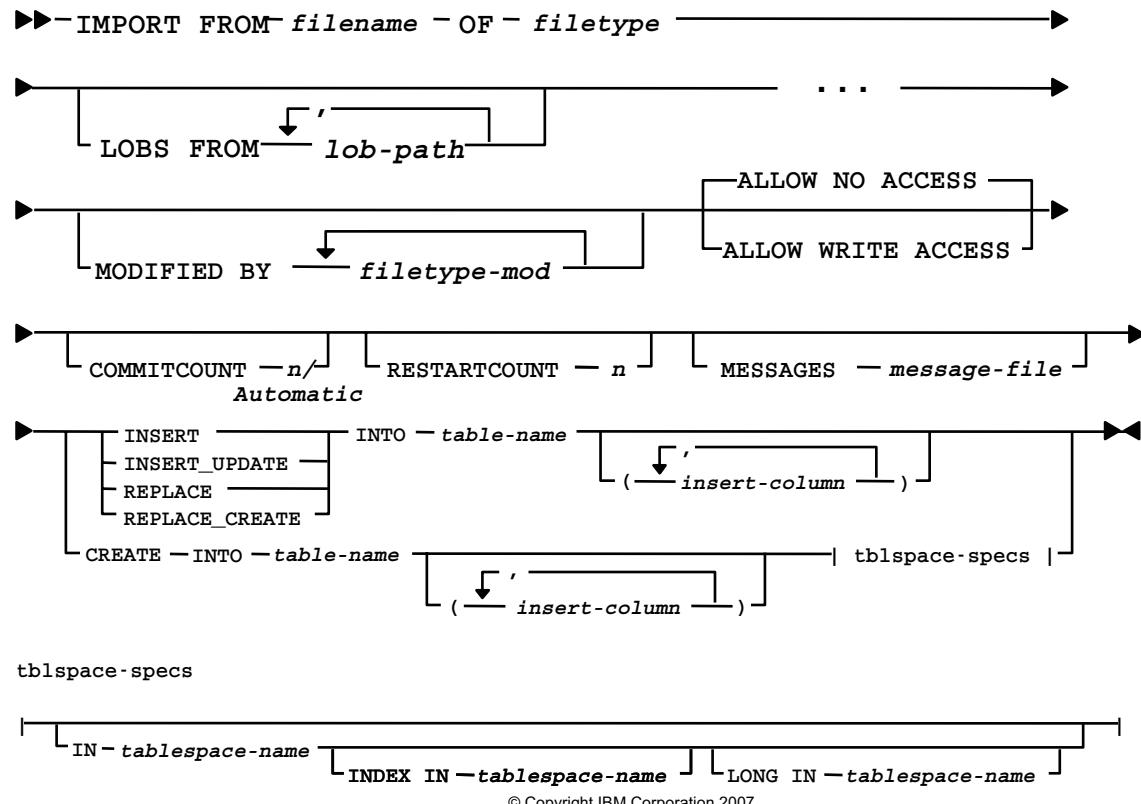


Figure 6-9. IMPORT command syntax (basic)

CF238.3

Notes:

The syntax of the IMPORT command is shown. More information on its options will be shown via examples on the following pages.

As default during the import an exclusive (X) lock is on the target table. This prevents concurrent applications from accessing table data. With the ALLOW WRITE ACCESS option the import runs in online mode. An intent exclusive (IX) is set on the target table. This allows concurrent readers and writers to access the table data. ALLOW WRITE ACCESS is not possible with the REPLACE, CREATE, or REPLACE_CREATE options, or with buffered inserts. The import operation will periodically commit inserted data to prevent lock escalation and to avoid running out of active log space. These commits will be performed even if the COMMITCOUNT option was not used.

The COMMITCOUNT n/AUTOMATIC performs a commit after every *n* records. When AUTOMATIC is specified, the import internally determines when a commit needs to be performed. The import utility will commit for either one of two reasons:

- To avoid running out of active log space

- To avoid lock escalation

If ALLOW WRITE ACCESS option is specified and the COMMITCOUNT option is not specified, the import utility will perform commits as if COMMITCOUNT AUTOMATIC has been specified.

Some IMPORT command parameters:

LOBS FROM lob-path

The names of the LOB data files are stored in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the LOB column. The maximum number of paths that can be specified is 999. This option is ignored if the lobsinfile modifier is not specified.

MODIFIED BY filetype-mod

Specifies file type modifier options.

compound=x	x is a number between 1 and 100 inclusive. Uses nonatomic compound SQL to insert the data, and x statements will be attempted each time.
generatedignore	This modifier informs the import utility that data for all generated columns is present in the data file but should be ignored. This results in all values for the generated columns being generated by the utility. This modifier cannot be used with the generatedmissing modifier.
generatedmissing	If this modifier is specified, the utility assumes that the input data file contains no data for the generated columns (not even NULLs), and will therefore generate a value for each row. This modifier cannot be used with the generatedignore modifier.
identityignore	This modifier informs the import utility that data for the identity column is present in the data file but should be ignored. This results in all identity values being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with the identitymissing modifier.
identitymissing	If this modifier is specified, the utility assumes that the input data file contains no data for the identity column (not even NULLs), and will therefore generate a value for each row. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This modifier cannot be used with the identityignore modifier.
lobsinfile	lob-path specifies the path to the files containing LOB data.
no_type_id	Valid only when importing into a single sub-table. Typical usage is to export data from a regular table, and then to invoke an

	import operation (using this modifier) to convert the data into a single sub-table.
nodefaults	If a source column for a target table column is not explicitly specified, and the table column is not nullable, default values are not loaded.
norowwarnings	Suppresses all warnings about rejected rows.
seclabelchar	Indicates that security labels in the input source file are in the string format for security label values rather than in the default encoded numeric format.
seclabelname	Indicates that security labels in the input source file are indicated by their name rather than the default encoded numeric format.
usedefaults	If a source column for a target table column has been specified, but it contains no data for one or more row instances, default values are loaded.

ALLOW NO ACCESS

Runs import in the offline mode. An exclusive (X) lock on the target table is acquired before any rows are inserted. This prevents concurrent applications from accessing table data. This is the default import behavior.

ALLOW WRITE ACCESS

Runs import in the online mode. An intent exclusive (IX) lock on the target table is acquired when the first row is inserted. This allows concurrent readers and writers to access table data.

COMMITCOUNT n/AUTOMATIC

Performs a COMMIT after every n records are imported. When a number n is specified, import performs a COMMIT after every n records are imported. When compound inserts are used, a user-specified commit frequency of n is rounded up to the first integer multiple of the compound count value. When AUTOMATIC is specified, import internally determines when a commit needs to be performed.

RESTARTCOUNT n

Specifies that an import operation is to be started at record n + 1. The first n records are skipped. This option is functionally equivalent to SKIPCOUNT. RESTARTCOUNT and SKIPCOUNT are mutually exclusive.

MESSAGES message-file

Specifies the destination for warning and error messages that occur during an import operation.

INSERT

Adds the imported data to the table without changing the existing table data.

INSERT_UPDATE

Adds rows of imported data to the target table, or updates existing rows (of the target table) with matching primary keys.

REPLACE

Deletes all existing data from the table by truncating the data object, and inserts the imported data. The table definition and the index definitions are not changed. This option can only be used if the table exists.

REPLACE_CREATE

If the table exists, deletes all existing data from the table by truncating the data object, and inserts the imported data without changing the table definition or the index definitions.

If the table does not exist, creates the table and index definitions, as well as the row contents, in the code page of the database.

INTO table-name

Specifies the database table into which the data is to be imported. This table cannot be a system table, a declared temporary table or a summary table.

CREATE

Creates the table definition and row contents in the code page of the database. If the data was exported from a DB2 table, sub-table, or hierarchy, indexes are created.

IN tablespace-name

Identifies the table space in which the table will be created. The table space must exist, and must be a REGULAR table space. If no other table space is specified, all table parts are stored in this table space. If this clause is not specified, the table is created in a table space created by the authorization ID.

INDEX IN tablespace-name

Identifies the table space in which any indexes on the table will be created. This option is allowed only when the primary table space specified in the IN clause is a DMS table space. The specified table space must exist, and must be a REGULAR or LARGE DMS table space.

LONG IN tablespace-name

Identifies the table space in which the values of any long columns (LONG VARCHAR, LONG VARGRAPHIC, LOB data types, or distinct types with any of these as source types) will be stored.

IMPORT command example

- Imports data from a file to a database table
- Check message for error or warning messages

```
db2 connect to musicdb
db2 import from artexprt of ixf
  messages artmsg
    create
    insert
    insert_update
    replace
    replace_create
.....
db2 import from artexprt of ixf modified by
compound 100
messages ...
```



© Copyright IBM Corporation 2007

Figure 6-10. IMPORT command example

CF238.3

Notes:

There are several IMPORT options available to load a table.

- **CREATE** — If a table does not exist already, the table is created using PC/IXF file information before the data is inserted.
- **INSERT** — Data is inserted into an existing table without changing the existing table data.
- **INSERT_UPDATE** — Add rows of imported data to the target table or updates existing rows (of the target table) with matching primary keys.
- **REPLACE** — Data in the existing table is deleted before the new data is inserted.
- **REPLACE_CREATE** — Old data in an existing table is deleted before the new data is inserted. If the specified table does not already exist, then the table is created using PC/IXF file information before the data is inserted.

MESSAGES with no message file specified will result in messages written to standard output.

COMMITCOUNT n/AUTOMATIC performs a commit every n records. This option can help to keep log sizes manageable. When AUTOMATIC is specified, the import internally determines when a commit needs to be performed. The import utility will commit for either of two reasons:

- To avoid running out of active log space
- To avoid lock escalation

If the ALLOW WRITE ACCESS option is specified and the COMMITCOUNT option is not specified, the import utility will perform commits as if COMMITCOUNT AUTOMATIC has been specified.

RESTARTCOUNT n specifies that an import is to be started at **record n+1**. The first n records are skipped.

METHOD L specifies the start and end column numbers from which to import data. This option must be used for ASC files.

METHOD N specifies the names of the columns to be imported.

METHOD P specifies the order of column numbers to be imported.

To import using INSERT, you must have SYSADM, DBADM, or CONTROL on the table, or INSERT and SELECT on the table.

To import to an existing table using INSERT_UPDATE, REPLACE, or REPLACE_CREATE, you must have SYSADM, DBADM, or CONTROL on the table.

To import to a table that does not exist using CREATE or REPLACE_CREATE, you must have SYSADM, DBADM, or CREATETAB authority and either IMPLICIT_SCHEMA or CREATEIN privilege.

When using the IMPORT utility:

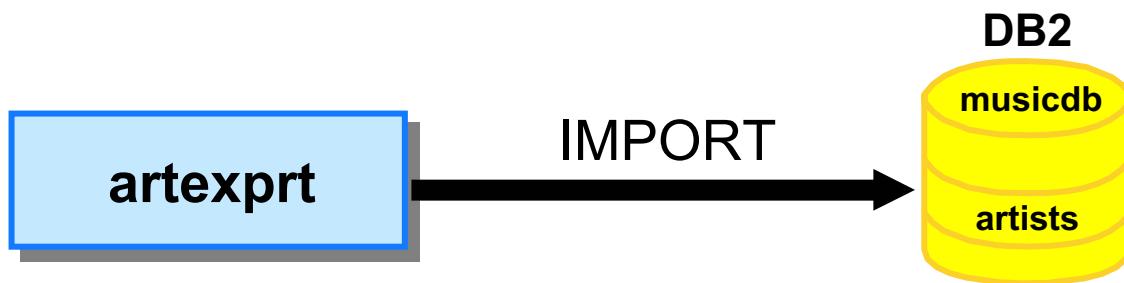
- REPLACE and REPLACE_CREATE functions are not allowed if the object table has any dependents other than itself.
- Only a simple "SELECT * FROM table" was used during EXPORT.
- The success of importing into a table with self-referencing constraints depends on the order in which the rows are inserted.

The MODIFIED BY option allows specification of additional information. Documentation of specific options is provided in the *Data Movement Utilities Guide and Reference* and the *Command Reference*. One such option is **COMPOUND n**, which allows n statements to be sent as a block using non-atomic compound SQL. The COMPOUND value ranges from 1 to 100. COMPOUND can greatly reduce the overhead associated with importing data.

Import command ... CREATE ... IN TABLESPACE

- Imports data from a file to three DMS table spaces

```
db2 connect to musicdb
db2 import from artexprt of ix
  messages artmsg  create into   artists [(column_list)]
    in           <tablespace>
    index        in <indextablespace>
    long         in <largertablespace>
```



© Copyright IBM Corporation 2007

Figure 6-11. Import command... CREATE... IN TABLESPACE

CF238.3

Notes:

You can specify a target table space when issuing an import with the CREATE option. Note that to specify a table space for indexes or long data, you must specify a table space for the table. All three table spaces must be DMS if an index or long table space is indicated. You can indicate that the table should be created in an SMS table space, but all three components will be stored in the same table space.

It is not required to specify a separate table space for long data or indexes.

6.2 LOAD utility

Four phases of Load



1. LOAD

Load data into tables
 Collect index keys / sort
 Consistency points at SAVECOUNT
 Invalid data rows in dump file; messages in message file



2. BUILD

Indexes created or updated

3. DELETE

Unique Key Violations placed in Exception Table
 Messages generated for unique key violations
 Deletes Unique Key Violation Rows

4. INDEX COPY

Copy indexes from temp table space to index table space

© Copyright IBM Corporation 2007

Figure 6-12. Four phases of Load

CF238.3

Notes:

The LOAD utility is faster than the IMPORT utility, because it writes formatted pages directly into the database, whereas the import utility performs SQL INSERTs. The LOAD utility does not fire triggers, and does not perform referential or table constraint checking (other than validating the uniqueness of the indexes). The LOAD utility also performs only minimal logging.

The LOAD utility is intended for the initial load or an append of a table where large amounts of data are to be moved. All data types (except XML) can be loaded by the LOAD utility, including large objects (LOBs) and user-defined types (UDTs).

The following file formats are supported with LOAD:

- Non-delimited ASCII
- Delimited ASCII
- PC/IXF (International Exchange Format)

There are four phases of the LOAD process:

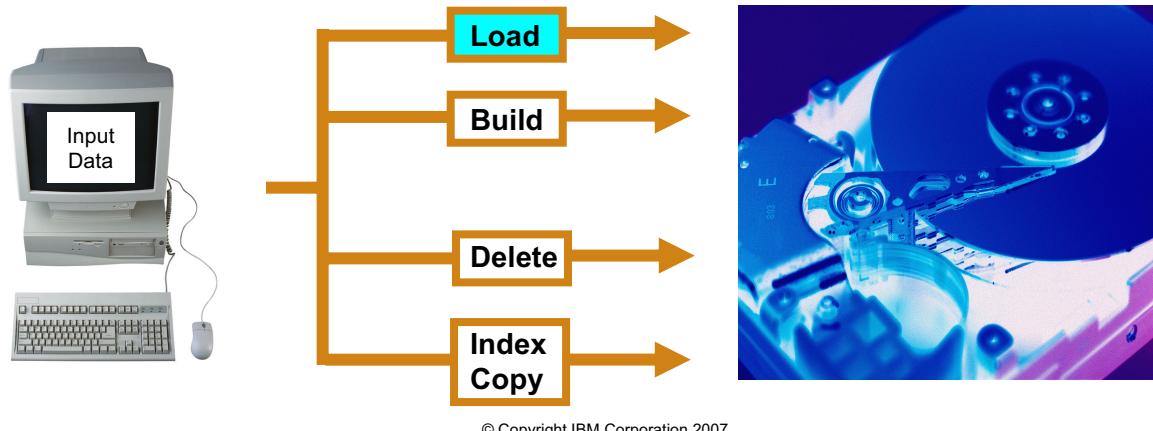
1. **Load** — when the data is inserted into the table.
2. **Build** — when the indexes are created or updated.
3. **Delete** — when the rows that caused a unique key violation are removed from the table.
4. **Index Copy** — when the index data is copied from a system temporary table space to the original table space.

The LOAD utility displays messages during the processing of each phase. If a failure occurs during the LOAD process, the messages will assist you in deciding how to recover from the failure. The LOAD utility can be invoked with a RESTART option, and the LOAD utility will determine where it needs to start processing.

The LOAD utility loads data from an input file, a device, or a pipe into a table. **This table must exist.** Indexes on the table may or may not exist; LOAD only builds indexes that are already defined on the table. If the table receiving the data already contains data, you can replace or append to the existing data.

LOAD: Load phase

- During the LOAD phase, data is stored in a table and index keys are collected
- Save points are established at intervals
- Messages indicate the number of input rows successfully loaded
- If a failure occurs in this phase, use the RESTART option for LOAD to restart from the last successful consistency point



© Copyright IBM Corporation 2007

Figure 6-13. LOAD: Load phase

CF238.3

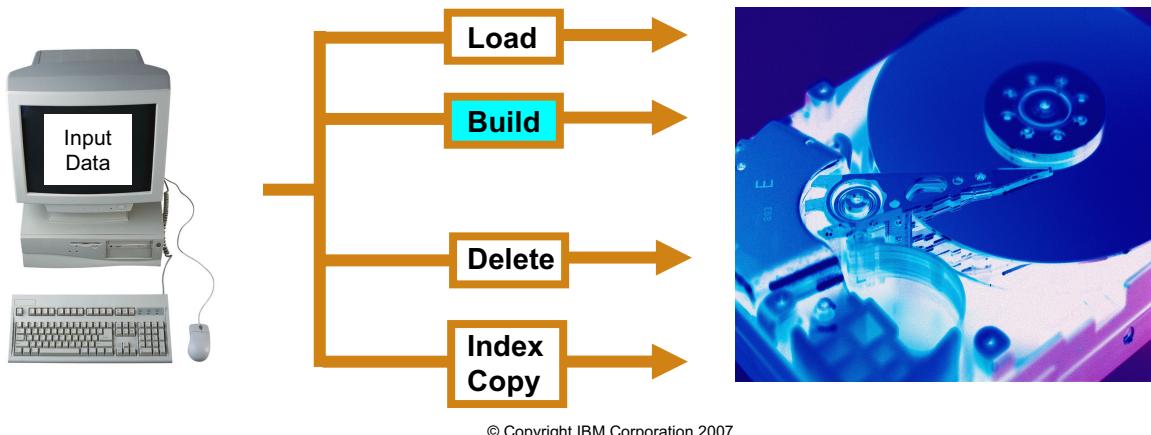
Notes:

During the **Load phase**, data is loaded into the table, and index keys and table statistics are collected, if necessary.

Save points, or *points of consistency*, are established at intervals specified by you in the **SAVECOUNT** parameter on the LOAD command. Messages are generated to let you know how many input rows have been successfully loaded at the time of the save point. If a failure occurs, you can restart the LOAD operation; the RESTART option automatically restarts the LOAD from the last successful consistency point. The TERMINATE option rolls back the failed load operation.

LOAD: 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 during this phase, LOAD restarts from the BUILD phase



© Copyright IBM Corporation 2007

Figure 6-14. LOAD: Build phase

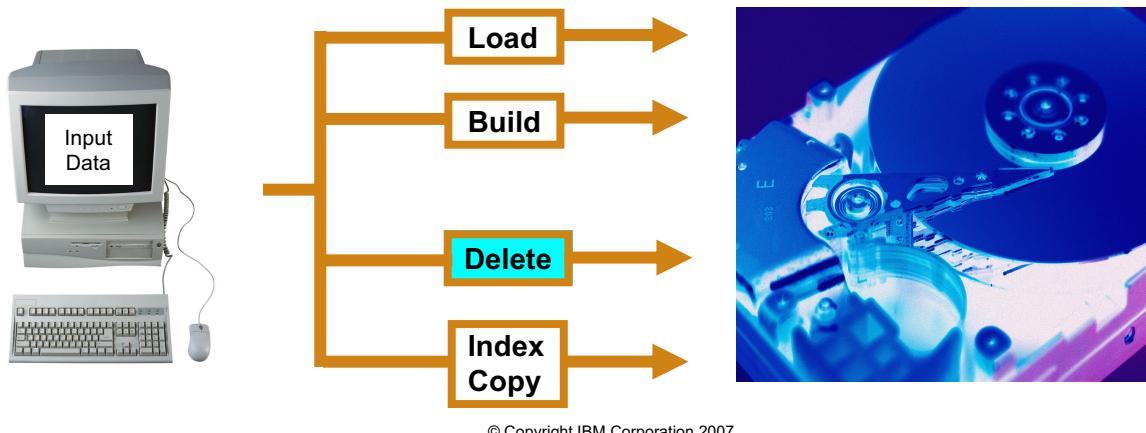
CF238.3

Notes:

During the **Build phase**, indexes are produced based on the index keys collected during the Load phase. The index keys are sorted during the load phase and index statistics are collected (if the STATISTICS YES with INDEXES option was specified). The statistics are similar to those collected through the RUNSTATS command. If a failure occurs during the Build phase, the RESTART option automatically restarts the load operation at the appropriate point.

LOAD: Delete phase

- During the DELETE phase, all rows that have violated a unique constraint are deleted
- If a failure occurs, LOAD restarts from the DELETE phase
- Once the database indexes are rebuilt, information about the rows containing the invalid keys is contained in an exception table, **WHICH SHOULD BE CREATED BEFORE LOAD**
- Finally, any duplicate keys found are deleted



© Copyright IBM Corporation 2007

Figure 6-15. LOAD: Delete phase

CF238.3

Notes:

During the **Delete phase**, rows that caused a unique key violation are removed from the table.

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

Do not attempt to delete or to modify any temporary files created by the load utility. Some temporary files are critical to the Delete phase. If a failure occurs during the Delete phase, the RESTART option automatically restarts the load operation at the appropriate point.

Each deletion event is logged. If you have a large number of records that violate the uniqueness condition, the log could fill up during the Delete phase.

LOAD: Index Copy phase

- The index data is copied from a system temporary table space to the original table space.
- This will only occur if a system temporary table space was specified for index creation during a load operation with the READ ACCESS option specified.

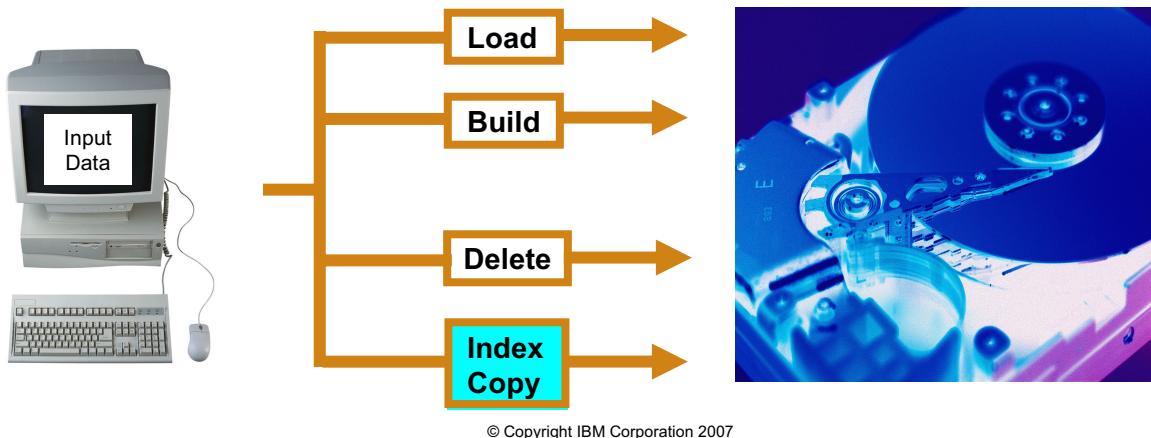


Figure 6-16. LOAD: Index Copy phase

CF238.3

Notes:

During the **Index copy** phase, the index data is copied from a system temporary table space to the original table space. This will only occur if a system temporary table space was specified for index creation during a load operation with the READ ACCESS option specified.

Load command

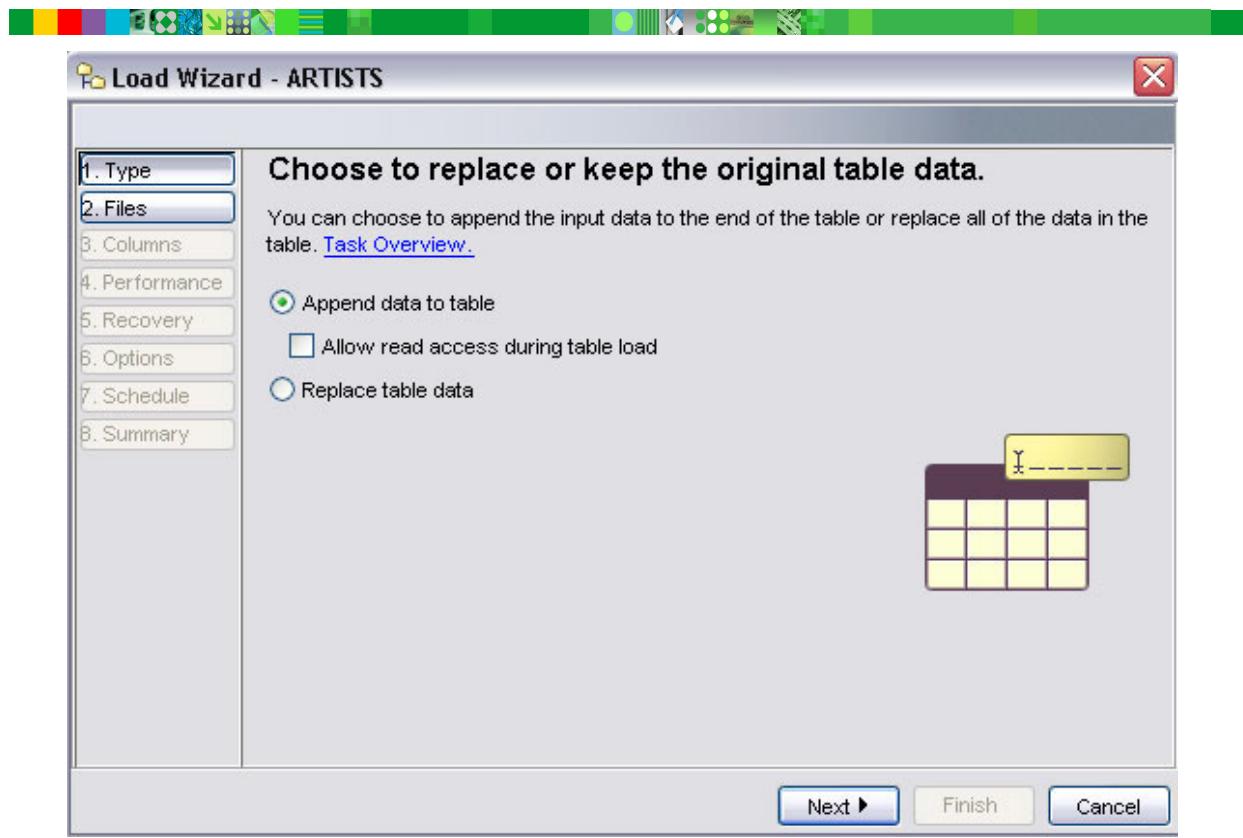


Figure 6-17. Load command

CF238.3

Notes:

To load data through the Load wizard:

- Left click and select **Tables**

Right-click and select **the table** that you plan to work with from the *Contents* pane

- Select **Load** — during the load, data is loaded from a file into a table.

You will find the following pages in the Load wizard:

- Type
- Files
- Columns
- Performance
- Recovery
- Options
- Schedule
- Integrity
- Summary

The Load GUI allows the method (append or replace) for the load operation to be specified. It allows data to be loaded from files, pipes, or devices to tables, and it allows specification of the messages file. It allows selection of columns to be loaded.

On the **Performance** page, you specify options such as the level of checking of the input data, how indexes are updated, how the load interacts with other applications, the percentage of each data page left free, and if you want a storage management snapshot (if you have storage management enabled).

Recovery options include creating consistency points, generating row count messages, the order of the source data, what actions to take if the load fails, and how to handle forward recovery.

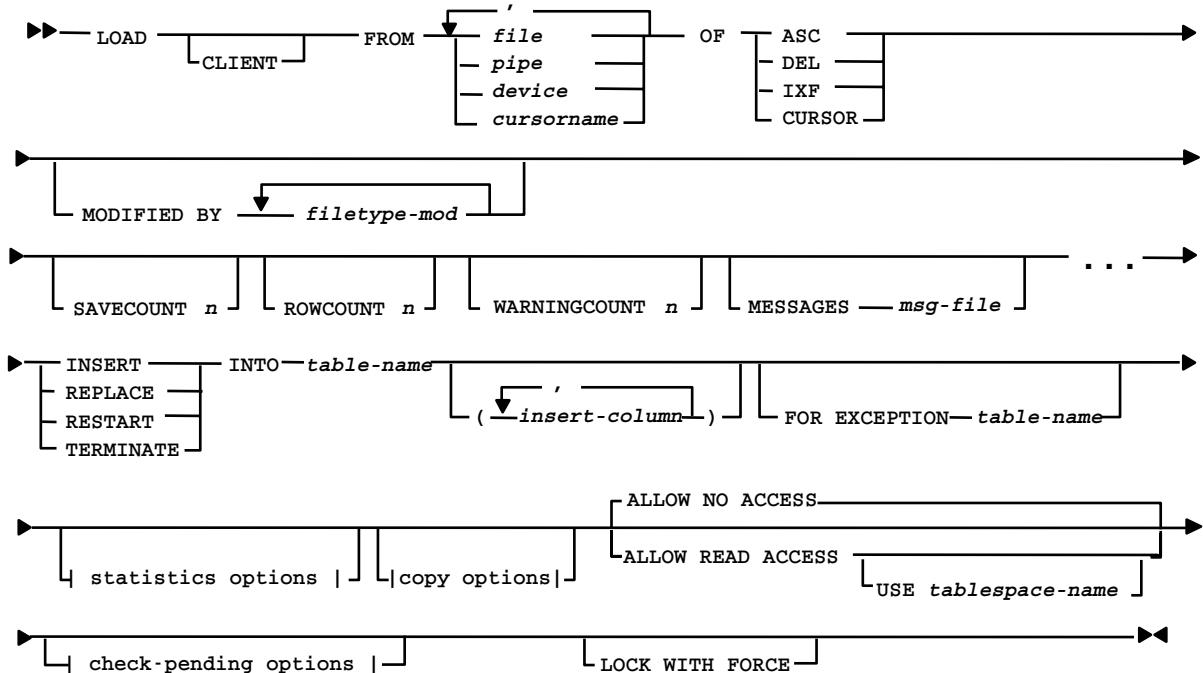
On the **Options** page, you can specify additional, more advanced options for the load task.

On the **Schedule** page, you specify when you want DB2 to perform the load task: you can run the task immediately, save the task to the Task Center, or schedule the task to be run at a later time.

If there are dependent tables on the table being loaded, the Integrity page will show a list of the tables that are dependent on the table being loaded; you can use this as an aid to choose the set integrity actions for the dependent tables after a successful load.

Finally, the **Summary** page will reflect the options you chose as you progressed through the wizard.

LOAD command syntax (basic)



© Copyright IBM Corporation 2007

Figure 6-18. LOAD command syntax (basic)

CF238.3

Notes:

The command syntax for the LOAD command is very extensive. Only some of the many parameters will be discussed in the following sections. Details regarding the LOAD command and all of its options can be found in the *Data Movement Utilities Guide and Reference*.

Some LOAD command parameters:

CLIENT

Specifies that the data to be loaded resides on a remotely connected client. This option is ignored if the load operation is not being invoked from a remote client. This option is ignored if specified in conjunction with the CURSOR filetype.

FROM filename/pipename/device/cursorname

Specifies the file, pipe, device, or cursor referring to an SQL statement that contains the data being loaded. If the input source is a file, pipe, or device, it must reside on the database partition where the database resides, unless the CLIENT option is specified.

OF filetype

Specifies the format of the data:

- **ASC** (non-delimited ASCII format)
- **DEL** (delimited ASCII format)
- **IXF** (integrated exchange format, PC version), exported from the same or from another DB2 table.
- **CURSOR** (a cursor declared against a SELECT or VALUES statement).

MODIFIED BY filetype-mod

Specifies file type modifier options. See File type modifiers for the Load utility.

SAVECOUNT n

Specifies that the load utility should set consistency points after every n rows. This value is converted to a page count, and rounded up to intervals of the extent size.

ROWCOUNT n

Specifies the number of n physical records in the file to be loaded. Allows a user to load only the first n rows in a file.

WARNINGCOUNT n

Stops the load operation after n warnings. Set this parameter if no warnings are expected, but verification that the correct file and table are being used is desired.

MESSAGES message-file

Specifies the destination for warning and error messages that occur during the load operation.

INSERT

One of four modes under which the load utility can execute. Adds the loaded data to the table without changing the existing table data.

REPLACE

One of four modes under which the load utility can execute. Deletes all existing data from the table, and inserts the loaded data. The table definition and index definitions are not changed.

RESTART

One of four modes under which the Load utility can execute. Restarts a previously interrupted load operation. The load operation will automatically continue from the last consistency point in the Load, Build, or Delete phase.

TERMINATE

One of four modes under which the load utility can execute. Terminates a previously interrupted load operation, and rolls back the operation to the point in time at which it started, even if consistency points were passed.

INTO table-name

Specifies the database table into which the data is to be loaded. This table cannot be a system table or a declared temporary table. An alias, or the fully qualified or unqualified table name can be specified.

FOR EXCEPTION table-name

Specifies the exception table into which rows in error will be copied. Any row that is in violation of a unique index or a primary key index is copied. The exception table must exist prior to LOAD.

ALLOW NO ACCESS

Load will lock the target table for exclusive access during the load. The table state will be set to Load In Progress during the load. ALLOW NO ACCESS is the default behavior. It is the only valid option for LOAD REPLACE.

ALLOW READ ACCESS

Load will lock the target table in a share mode. The table state will be set to both Load In Progress and Read Access. Readers can access the non-delta portion of the data while the table is being load.

USE tablespace-name

If the indexes are being rebuilt, a shadow copy of the index is built in table space tablespace-name and copied over to the original table space at the end of the load during an INDEX COPY PHASE.

LOCK WITH FORCE

The utility acquires various locks including table locks in the process of loading. Rather than wait, and possibly timeout, when acquiring a lock, this option allows load to force off other applications that hold conflicting locks on the target table. Applications holding conflicting locks on the system catalog tables will not be forced off by the load utility. Forced applications will roll back and release the locks the load utility needs.

LOAD scenario

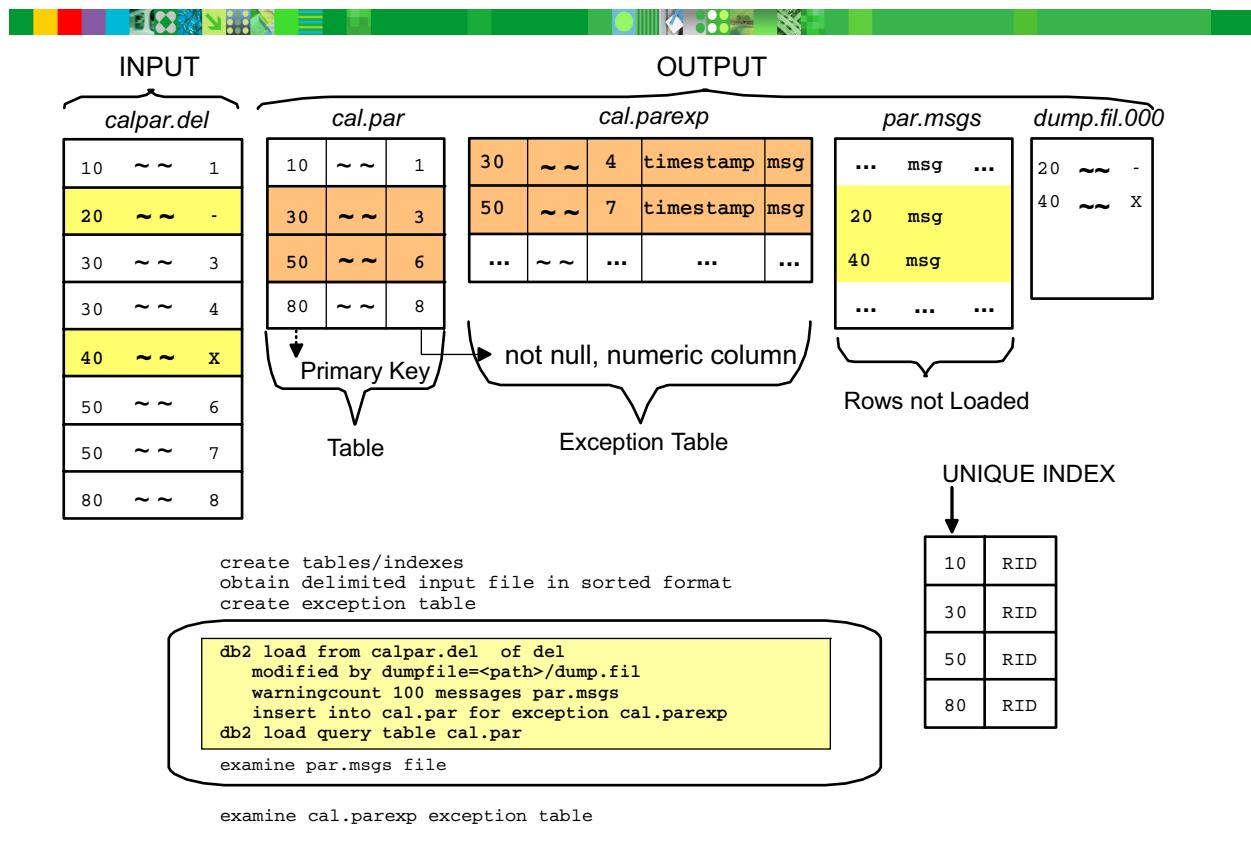


Figure 6-19. LOAD scenario

CF238.3

Notes:

In our LOAD scenario, we have created tables and indexes, sorted the input data (or obtained it in sorted order), and created the exception table.

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

FOR EXCEPTION indicates that any row that is in violation of a unique index rule will be stored in the table indicated. In our example, *cal.parexp* will contain those rows. If an exception table is not provided for the LOAD, and duplicate records are found, then the LOAD will continue. However, only a warning message is issued about the deletion of duplicate records, and the deleted duplicate records are not placed anywhere.

Other types of errors (for example, attempting to load a null into a column that is defined as NOT NULL) will cause a message to be written to the messages file. The second row in our example will cause this kind of warning in the message file. The DUMPFILE=qualified-filename option will write any rejected row to the named file. The

name must be a fully qualified name on the server. The file will be created with a partition number at the end; on a single partition database, this will be 000.

The exception tables used with LOAD are identical to the ones used in the SET INTEGRITY statement. They can be reused during checking with the SET INTEGRITY statements. There are a number of rules for creating exception tables; we will see them on the next visual.

In our example, cal is the owner/schema of the table.

To load data into a table, you must have one of the following:

- SYSADM or DBADM authority
- LOAD privilege on the database, and
 - INSERT privilege on the table when the load utility is invoked in INSERT mode
 - INSERT and DELETE privilege on the table when the load utility is invoked in REPLACE mode
 - INSERT privilege on the exception table, if used

Since you will typically be loading large amounts of data using the LOAD command, a LOAD QUERY command can be used to check the progress of the LOAD process. Options on the LOAD QUERY command allow you to indicate that you only want to see summary information (SUMMARYONLY), or just the new information since the last LOAD QUERY was issued (SHOWDELTA).

Rules for creating Exception Table

- The first n columns are the same
- No constraints and no trigger definitions
- n+1 column **TIMESTAMP**
- n+2 column **CLOB (32 KB)**
- user **INSERT privilege**

© Copyright IBM Corporation 2007

Figure 6-20. Rules for creating Exception Table

CF238.3

Notes:

The first **n** columns of the exception table reflect the definition of the table being loaded. All column attributes (type, length, and nullability attributes) should be identical. An exception table may not contain an identity column or any other type of generated column.

All the columns of the exception table should be free of any constraints and triggers. Constraints include referential integrity and check constraints, as well as unique index constraints that could cause errors on insert.

The **n+1** column of the exception table is an optional **TIMESTAMP** column. The timestamp column in the exception table may be used to distinguish newly-inserted rows from the old ones, if necessary.

The **n+2** column should be of type **CLOB (32 KB)** or larger. This column is optional but recommended, and will be used to give the names of the constraints that the data within the row violates.

No additional columns are allowed.

If the original table has generated columns (including the IDENTITY property), the corresponding columns in the exception table should not specify the generated property.

It should also be noted that a user invoking any facility (LOAD, SET INTEGRITY) that may cause rows to be inserted into the exception table must have INSERT privilege on the exception table.

For format of the Message Column Structure:

1. Number of constraint violations
2. Type of first constraint violation
3. Length of constraint/index name
4. Constraint name/index name
5. Separator
6. Type of next constraint violation
7. Length of constraint/index name
8. Constraint name/index name

Fields 5-8 repeat for each violation.

Suggested methods for creating Exception Tables



– CREATE TABLE T1EXC LIKE T1

```
ALTER TABLE T1EXC
  ADD COLUMN TS TIMESTAMP
  ADD COLUMN MSG CLOB(32K)
```

– CREATE TABLE T1EXC AS
 (SELECT T1.*,
 CURRENT TIMESTAMP AS TS,
 CLOB(' ', 32767) AS MSG
 FROM T1)
 DEFINITION ONLY

© Copyright IBM Corporation 2007

Figure 6-21. Suggested methods for creating Exception Tables

CF238.3

Notes:

There are several methods that can be used to create the exception tables. Two options are shown on the graphic that will create an exception table t1exc that could be used as an exception table for table t1. Two additional options would include:

- Generate DDL using the Control Center

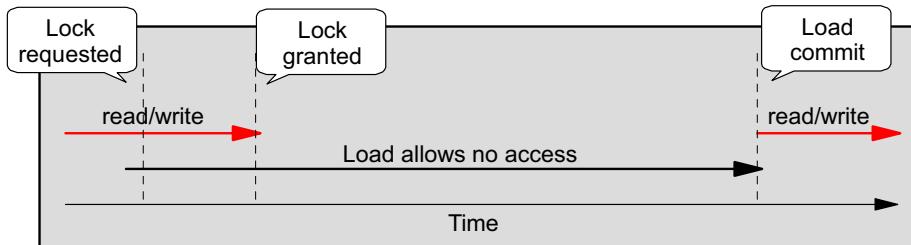
The Control Center gives you the option of performing reverse engineering to create the DDL for a table. You can use this option to obtain the DDL for the original table, edit the statement generated to add the additional columns, and then run the file.

- Export using IXF

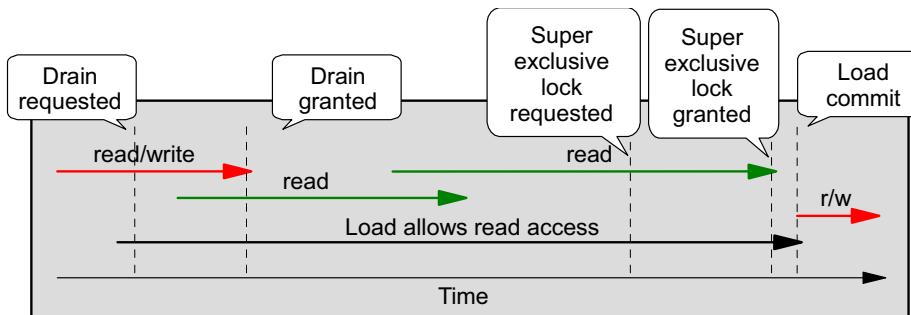
Since the IXF format of data carries the definition of the table as well as any data selected, you can use it to capture the definition of a table. The IMPORT... CREATE option would be used to create the new table, which would then be altered to add the additional columns.

Offline versus Online Load

- ALLOW NO ACCESS



- ALLOW READ ACCESS



© Copyright IBM Corporation 2007

Figure 6-22. Offline versus Online Load

CF238.3

Notes:

In most cases, the LOAD utility uses table-level locking to restrict access to tables. The LOAD utility does not quiesce the table spaces involved in the load operation, and uses table space states only for load operations with the COPY NO option specified. The level of locking depends on whether the load operation allows read access. A load operation in ALLOW NO ACCESS mode will use an exclusive lock (Z-lock) on the table for the duration of the load. An load operation in ALLOW READ ACCESS mode acquires and maintains a share lock (S-lock) for the duration of the load operation, and upgrades the lock to an exclusive lock (Z-lock) when data is being committed.

Before a load operation in ALLOW READ ACCESS mode begins, the load utility will wait for all applications that began before the load operation to release locks on the target table. Since locks are not persistent, they are supplemented by table states that will remain even if a load operation is aborted. These states can be checked by using the LOAD QUERY command. By using the LOCK WITH FORCE option, the LOAD utility will force applications holding conflicting locks off the table into which it is trying to load.

ALLOW NO ACCESS is the default option on the LOAD utility.

Locking Behavior for Load Operations in ALLOW READ ACCESS Mode

At the beginning of a load operation, the LOAD utility acquires a share lock (S-lock) on the table. It holds this lock until the data is being committed. The share lock allows applications with compatible locks to access the table during the load operation. For example, applications that use read-only queries will be able to access the table, while applications that try to insert data into the table will be denied. When the LOAD utility acquires the share lock on the table, it will wait for all applications that hold locks on the table prior to the start of the load operation to release them, even if they have compatible locks.

Since the LOAD utility upgrades the share lock to an exclusive lock (Z-lock) when the data is being committed, there may be some delay in commit time while the LOAD utility waits for applications with conflicting locks to finish.

Note: The load operation will not time out while it waits for the applications to release their locks on the table.

LOCK WITH FORCE Option

The LOCK WITH FORCE option can be used to force off applications holding conflicting locks on a table so that the load operation can proceed. If an application is forced off the system by the load utility, it will lose its database connection and an error will be returned.

For a load operation in ALLOW NO ACCESS mode, all applications holding table locks that exist at the start of the load operation will be forced.

For a load operation in ALLOW READ ACCESS mode, applications holding the following locks will be forced:

- Table locks that conflict with a table share lock (for example, import or insert).
- All table locks that exist at the commit phase of the load operation.

When the COPY NO option is specified for a load operation on a recoverable database, all objects in the target table space will be locked in share mode before the table space is placed in backup pending state. This will occur regardless of the access mode. If the LOCK WITH FORCE option is specified, all applications holding locks on objects in the table space that conflict with a share lock will be forced off.

Table states



- (Load pending, Set Integrity Pending)
 - LOAD QUERY TABLE <table-name>
 - Tablestate:
 - Normal
 - Set Integrity Pending
 - Load in Progress
 - Load Pending
 - Reorg Pending
 - Read Access Only
 - Unavailable
 - Not Load Restartable
 - Unknown
 - Table can be in several states at same time
- Tablestate:**
- Set Integrity Pending
 - Load in Progress
 - Read Access Only

© Copyright IBM Corporation 2007

Figure 6-23. Table states

CF238.3

Notes:

In addition to locks, the LOAD utility uses table states to control access to tables. A table state can be checked by using the LOAD QUERY command. The states returned by the LOAD QUERY command are as follows:

- **Normal** — No table states affect the table.
- **Set Integrity Pending** — The table has constraints which have not yet been verified. Use the SET INTEGRITY statement to take the table out of Check Pending state. The LOAD utility places a table in Set Integrity Pending state when it begins a load operation on a table with constraints.
- **Load in Progress** — There is a load operation in progress on this table.
- **Load Pending** — A load operation has been active on this table but has been aborted before the data could be committed. Issue a LOAD TERMINATE, LOAD RESTART, or LOAD REPLACE command to bring the table out of this state.

- **Read Access Only** — The table data is available for read access queries. Load operations using the ALLOW READ ACCESS option place the table in read access only state.
- **Reorg Pending** — There is a reorg operation in progress on this table.
- **Unavailable** — The table is unavailable. The table may only be dropped or restored from a backup. Rolling forward through a non-recoverable load operation will place a table in the unavailable state.
- **Not Load Restartable** — The table is in a partially loaded state that will not allow a load restart operation. The table will also be in load pending state. Issue a LOAD TERMINATE or a LOAD REPLACE command to bring the table out of the Not Load Restartable state. A table is placed in Not Load Restartable state when a roll forward operation is performed after a failed load operation that has not been successfully restarted or terminated, or when a restore operation is performed from an online backup that was taken while the table was in Load in Progress or Load Pending state. In either case, the information required for a load restart operation is unreliable, and the Not Load Restartable state prevents a load restart operation from taking place.
- **Unknown** — The LOAD QUERY command is unable determine the table state.

A table can be in several states at the same time. For example, if data is loaded into a table with constraints and the ALLOW READ ACCESS option is specified, the table state would be:

Tablestate:
Set Integrity Pending
Load in Progress
Read Access Only

After the load operation but before issuing the SET INTEGRITY statement, the table state would be:

Tablestate:
Set Integrity Pending
Read Access Only

After the SET INTEGRITY statement has been issued, the table state would be:

Tablestate:
Normal

Load Pending state: Recovering from LOAD failure



- Restart the Load
 - Check Messages files
 - Use Restart option
 - Load operation automatically continues from last consistency point in Load or Build phase
 - or Delete phase if ALLOW NO ACCESS
- Replace the whole table
 - `LOAD ... REPLACE`
- Terminate the Load
 - If `LOAD ... INSERT`, returns table to state preceding Load
 - If `LOAD ... REPLACE`, table will be truncated to an empty state
 - Create backup copy prior to Load to return to state preceding Load

Do not tamper with Load temporary files

© Copyright IBM Corporation 2007

Figure 6-24. Load Pending state: Recovering from LOAD failure

CF238.3

Notes:

If the LOAD utility cannot start because of a user error, such as a nonexistent data file or invalid column names, it will terminate and leave the table in a normal state.

If a failure occurs while loading data, you can restart the load operation from the last consistency point (using the RESTART option), reload the entire table (using the REPLACE option), or terminate the load (using the TERMINATE option). Specify the same parameters as in the previous invocation so that the utility can find the necessary temporary files.

A load operation that specified the ALLOW READ ACCESS option can be restarted using either the ALLOW READ ACCESS option or the ALLOW NO ACCESS option. A load operation that specified the ALLOW NO ACCESS option can only be restarted using the ALLOW NO ACCESS option. If the index object is unavailable or marked invalid, a load restart or terminate in ALLOW READ ACCESS mode will not be permitted. If the original load operation was aborted in the Index Copy phase, a restart operation in the ALLOW READ ACCESS mode is not permitted because the index may be corrupted.

If a load operation in ALLOW READ ACCESS mode was aborted in the Load phase, it will restart in the Load phase. If it was aborted in any phase other than the Load phase, it will restart in the Build phase. If the original load operation was in ALLOW NO ACCESS mode, a restart operation may occur in the Delete phase if the original load operation reached that point and the index is valid. If the index is marked invalid, the Load utility will restart the load operation from the Build phase.

Load REPLACE deletes all existing data from the table and inserts the loaded data. Using load REPLACE and specifying an empty input file will truncate the table.

Terminating the load will roll back the operation to the point in time at which it started, even if consistency points were passed. The states of any tables involved in the operation return to normal, and all table objects are made consistent (index objects may be marked as invalid, in which case index rebuild will automatically take place at the next access). If the load operation being terminated is a load REPLACE, the table will be truncated to an empty table after the load TERMINATE operation. If the load operation being terminated is a load INSERT, the table will retain all of its original records after the load TERMINATE operation.



Note

The Load operation writes temporary files onto the server into a subdirectory of the database directory by default (the location can be changed using a TEMPFILES PATH temp-pathname parameter on the LOAD command). The temporary files written to this path are removed when the load operation completes without error. These temporary files must not be tampered with under any circumstances. Doing so will cause the load operation to malfunction, and will place your database in jeopardy.

Backup Pending state — COPY options

- When loading data and forward recovery enabled:

- **COPY NO (default)**
 - During load, "Backup pending" and "Load in progress"
 - After load, "Backup Pending"
- **COPY YES**
 - Load has made Copy → not "Backup pending"
- **NONRECOVERABLE**
 - No copy made and no backup required

UNIX

→ Databasealias.Type.InstanceName.NodeName.CatNodeName.Timestamp.number

Windows

→ Databasealias.Type.InstanceName.NodeName.CatNodeName.Timestamp.number

Type: 0=Full Backup
3=Table Space Backup
4 =Copy from Table Load

© Copyright IBM Corporation 2007

Figure 6-25. Backup Pending state — COPY options

CF238.3

Notes:

If a load operation with the COPY NO option is executed in a recoverable database, the table spaces associated with the load operation are placed in the *Backup pending* table space state and the *Load in progress* table space state. This takes place at the beginning of the load operation. The load operation may be delayed at this point while locks are acquired on the tables within the table space.

When a table space is in Backup pending state, it is still available for read access. The table space can only be taken out of Backup pending state by taking a backup of the table space. Even if the load operation is aborted, the table space will remain in Backup pending state because the table space state is changed at the beginning of the load operation, and cannot be rolled back if it fails. The Load in Progress table space state prevents online backups of a load operation with the COPY NO option specified while data is being loaded. The Load in Progress state is removed when the load operation is completed or aborts.

Note that if the database is a recoverable database, then terminating the load will not eliminate the requirement to make a backup of the loaded table space.

When loading data, if forward recovery is enabled, there are three options to consider:

1. **COPY NO** (default) specifies that the table space in which the table resides will be placed in backup pending state if forward recovery is enabled (that is, logretain or userexit is on). COPY NO will also put the table space state into the Load in Progress table space state. This is a transient state that will disappear when the load completes or aborts. The data in any table in the table space cannot be updated or deleted until a table space backup or a full database backup is made. However, it is possible to access the data in any table by using the SELECT statement.
2. **COPY YES** on the LOAD specifies that a copy of the changes made will be saved. This copy is used during roll-forward recovery to recreate the changes to the database done by LOAD. This option is invalid if forward recovery is disabled. COPY YES slows the LOAD utility. If you are loading a lot of tables, you may wish to load all of the tables in the table space, then back it up.
3. **NONRECOVERABLE** specifies that the load transaction is to be marked as non-recoverable and that it will not be possible to recover it by a subsequent roll forward action. The roll forward utility will skip the transaction and will mark the table into which data was being loaded as *invalid*. The utility will also ignore any subsequent transactions against that table. After the roll forward operation is completed, such a table can only be dropped or restored from a backup (full or table space) taken after a commit point following the completion of the non-recoverable load operation.

With this option, table spaces are not put into backup pending state following the load operation, and a copy of the loaded data does not have to be made during the load operation. This can be used to enable loading several tables in a table space before performing a backup of the table space. It can also be used with tables that are always loaded with LOAD... REPLACE since they could be recovered by reloading.

If you do not create a copy, the LOAD will execute more quickly. You must also allow for the disk space that an additional backup copy would take.

The name of the backup file has entries for the **Type** field to indicate what it represents. There are three options:

1. 0 for full database backup
2. 3 for table space backup
3. 4 for Copy from Table Load

If load is used with the **nonrecoverable** option, there is no requirement to take a backup.

During a roll forward operation through a LOAD command with the COPY NO option specified, the associated table spaces are placed in *Restore pending* state. To remove the table spaces from *Restore pending* state, a restore operation must be performed. A roll forward operation will only place a table space in the *Restore pending* state if the load operation completed successfully.

Set Integrity Pending table state



- Load turns OFF constraint checking
 - Leaves table in Set Integrity Pending state
 - If parent table is in Set Integrity Pending, then dependents may also be in Set Integrity Pending
 - **LOAD INSERT, ALLOW READ ACCESS**
 - Loaded table in Set Integrity Pending with read access
 - **LOAD INSERT, ALLOW NO ACCESS**
 - Loaded table in Set Integrity Pending with no access
 - **LOAD REPLACE, SET INTEGRITY PENDING CASCADE DEFERRED**
 - Loaded table in Set Integrity Pending with no access
 - **LOAD REPLACE, SET INTEGRITY PENDING CASCADE IMMEDIATE**
 - Loaded table and descendant foreign key tables are in Set Integrity Pending with no access

© Copyright IBM Corporation 2007

Figure 6-26. Set Integrity Pending table state

CF238.3

Notes:

Following a load operation, the loaded table may be in set integrity pending state in either READ or NO ACCESS mode if the table has table check constraints or referential integrity constraints defined on it. If the table has descendant foreign key tables, they may also be in set integrity pending state.

If the loaded table has descendant tables, the SET INTEGRITY PENDING CASCADE parameter can be specified to indicate whether the check pending state of the loaded table should be immediately cascaded to the descendant materialized query tables or descendant staging tables. SET INTEGRITY PENDING CASCADE does not apply to descendant foreign key tables. If the loaded table has constraints as well as descendant foreign key tables, and if all of the tables are in normal state prior to the load operation, the following will result based on the load parameters specified:

INSERT, ALLOW READ ACCESS, and SET INTEGRITY PENDING CASCADE IMMEDIATE or DEFERRED

The loaded table will be placed in Set Integrity Pending state with read access. Descendent foreign key tables will remain in their original states.

INSERT, ALLOW NO ACCESS, and SET INTEGRITY PENDING CASCADE IMMEDIATE

The loaded table will be placed in Set Integrity Pending state with no access. Descendent foreign key tables will remain in their original states.

INSERT or REPLACE, ALLOW NO ACCESS, and SET INTEGRITY PENDING CASCADE DEFERRED

Only the loaded table will be placed in Set Integrity Pending state with no access. Descendent foreign key tables will remain in their original states.

REPLACE, ALLOW NO ACCESS, and SET INTEGRITY PENDING CASCADE IMMEDIATE

The table and all its descendent foreign key tables will be placed in Set Integrity Pending state with no access.

**Note**

Specifying the ALLOW READ ACCESS option in a load replace operation will result in an error.

SET INTEGRITY command overview



- SET INTEGRITY is used to:
 - Bring one or more tables out of Set Integrity Pending state (previously known as *Check Pending state*)
 - Bring one or more tables out of Set Integrity Pending state without performing required integrity processing on those tables
 - Place one or more tables in Set Integrity Pending state
 - Place one or more tables into full access state
 - Prune the contents of one or more staging tables

© Copyright IBM Corporation 2007

Figure 6-27. SET INTEGRITY command overview

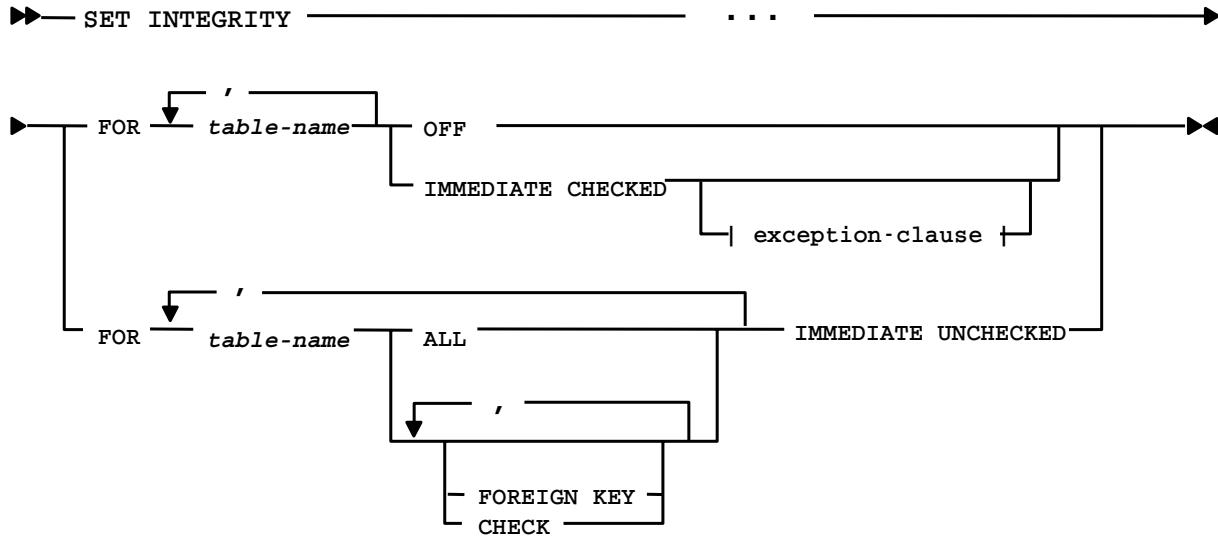
CF238.3

Notes:

When the statement is used to perform integrity processing for a table after it has been loaded or attached, the system can incrementally process the table by checking only the appended portion for constraints violations. If the subject table is a materialized query table or a staging table, and load, attach, or detach operations are performed on its underlying tables, the system can incrementally refresh the materialized query table or incrementally propagate to the staging table with only the delta portions of its underlying tables. However, there are some situations in which the system will not be able to perform such optimizations and will instead perform full integrity processing to ensure data integrity. Full integrity processing is done by checking the entire table for constraints violations, recomputing a materialized query table's definition, or marking a staging table as inconsistent. The latter implies that a full refresh of its associated materialized query table is required. There is also a situation in which you might want to explicitly request incremental processing by specifying the INCREMENTAL option.

The SET INTEGRITY statement is under transaction control.

SET INTEGRITY command syntax (basic)



exception-clause

```

    FOR EXCEPTION [table-name] IN [table-name] USE [table-name]
  
```

© Copyright IBM Corporation 2007

Figure 6-28. SET INTEGRITY command syntax ((basic))

CF238.3

Notes:

The **OFF** option specifies that the tables (or table) are to have their foreign key constraints and check constraints turned off, and are therefore to be placed into the set integrity pending state, where only limited access by a restricted set of statements and commands is allowed. Primary key and unique constraints continue to be checked.

The **IMMEDIATE CHECKED** option specifies that the table is to have its integrity checking turned on and that the integrity checking that was deferred is to be carried out. This is done in accordance with the information set in the STATUS and CONST_CHECKED columns of the SYSCAT.TABLES. The value in STATUS must be C (the table is in the set integrity pending state), and the value in CONST_CHECKED indicates which integrity options are to be checked.

The **IMMEDIATE UNCHECKED** option specifies that the table is to have its integrity checking turned on (and thus is to be taken out of the set integrity pending state) without having the table checked for integrity violations. The implications with respect to data integrity should be considered before using this option.

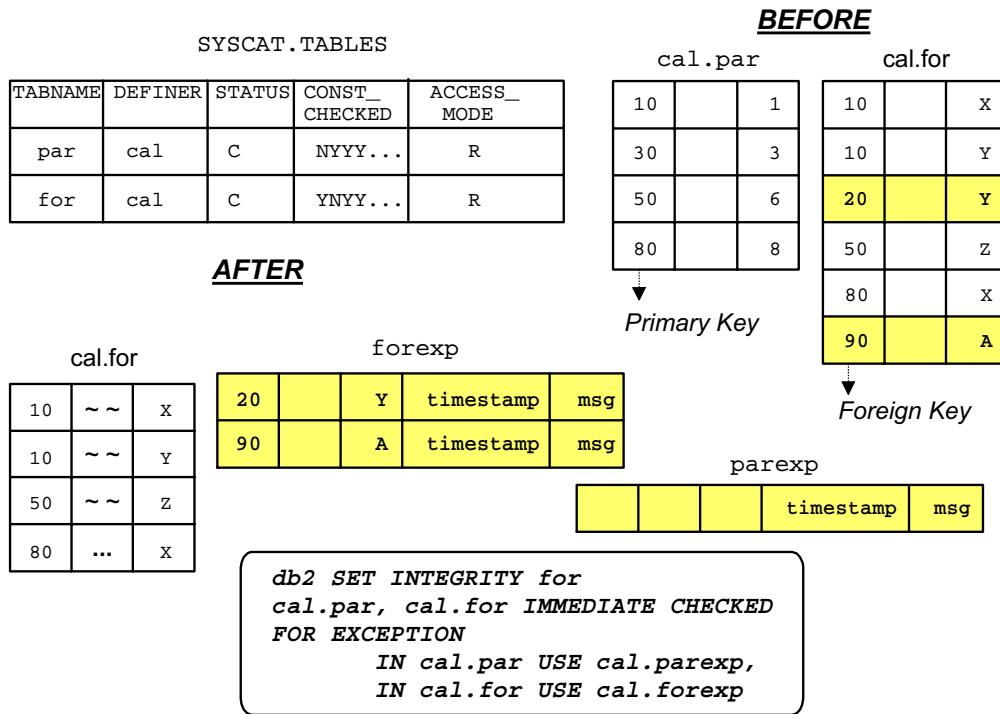
Sample query to check the pending states on the tables:

```
select tabname,
       substr(const_checked,1,1) as FK_CHECKED,
       substr(const_checked,2,1) as CC_CHECKED
  from syscat.tables
 where status='C'
```

Byte 1 of CONST_CHECKED column in SYSCAT.TABLES represents foreign key constraints. Byte 2 represents check constraints. Other bytes are reserved. Values are:

- **Y**=Checked by system
- **U**=Checked by user
- **N**=Not checked (pending)

Set Integrity Pending state



© Copyright IBM Corporation 2007

Figure 6-29. Set Integrity Pending state

CF238.3

Notes:

The STATUS flag of the SYSCAT.TABLES entry corresponding to the loaded table indicates the set integrity pending state of the table. For the loaded table to be fully usable, the STATUS must have a value of N and the ACCESS MODE must have a value of F, indicating that the table is in normal state and fully accessible.

In the example on the graphic, both CAL.PAR and CAL.FOR have been loaded. They both indicate that constraints need to be checked (STATUS = 'C'). CAL.PAR is a parent table, and CAL.FOR is its dependent. CAL.FOR also has a check constraint defined on one of its columns. In this case, both tables are in set integrity pending status because they have both been loaded.

The SET INTEGRITY statement is executed to check the constraints on both tables. CAL.FOR had two rows that did not have parent keys in CAL.PAR, so those rows were moved to the FOREXP exception table. The CAL.PAR table did not have any check constraint violations, so there are no rows added to the PAREXP table as a result of this SET INTEGRITY statement.

To remove the set integrity pending state, use the SET INTEGRITY statement. The SET INTEGRITY statement checks a table for constraint violations, and takes the table out of set integrity pending state. If all the load operations are performed in INSERT mode, the SET INTEGRITY statement can be used to incrementally process the constraints (that is, it will check only the appended portion of the table for constraint violations). For example:

```
db2 load from infile1.ixf of ixf insert into table1  
db2 set integrity for table1 immediate checked
```

Only the appended portion of TABLE1 is checked for constraint violations. Checking only the appended portion for constraint violations is faster than checking the entire table, especially in the case of a large table with small amounts of appended data.

If a table is loaded with the SET INTEGRITY PENDING CASCADE DEFERRED option specified, and the SET INTEGRITY statement is used to check for integrity violations on the parent table, the descendant tables will be placed in set integrity pending state with no access when the SET INTEGRITY statement is issued. To take the descendant tables out of this state, you must issue an explicit SET INTEGRITY request on the descendant tables.

If the ALLOW READ ACCESS option is specified for a load operation, the table will remain in read access state until the SET INTEGRITY statement is used to check for constraint violations. Applications will be able to query the table for data that existed prior to the load operation once it has been committed, but will not be able to view the newly loaded data until the SET INTEGRITY statement has been issued.

Several load operations can take place on a table before checking for constraint violations. If all of the load operations are completed in ALLOW READ ACCESS mode, only the data that it existed in the table prior to the first load operation will be available for queries.

One or more tables can be checked in a single invocation of this statement. If a dependent table is to be checked on its own, the parent table cannot be in set integrity pending state. Otherwise, both the parent table and the dependent table must be checked at the same time. In the case of a referential integrity cycle, all the tables involved in the cycle must be included in a single invocation of the SET INTEGRITY statement. It may be convenient to check the parent table for constraint violations while a dependent table is being loaded. This can only occur if the two tables are not in the same table space.

Use the load exception table option to capture information about rows with constraint violations.

The SET INTEGRITY statement does not activate any DELETE triggers as a result of deleting rows that violate constraints, but once the table is removed from set integrity pending state, triggers are active. Thus, if we correct data and insert rows from the exception table into the loaded table, any INSERT triggers defined on the table will be activated. The implications of this should be considered. One option is to drop the INSERT trigger, insert rows from the exception table, and then recreate the INSERT trigger.

Meaningful steps for LOAD



- Create tables and indexes
- Create exception tables
- Sort data
- Back up TS/DB (if using REPLACE)
- Consider freespace
- Load for Exception ... Savecount ... Warningcount...
- Examine xx.msg and dumpfile (after LOAD completes)
- Examine exception tables (after LOAD completes)
- Back up table space if log retain=recovery and COPY NO
- Set Integrity for (only if table in Check Pending state)
- Update statistics (if necessary)

© Copyright IBM Corporation 2007

Figure 6-30. Meaningful steps for LOAD

CF238.3

Notes:

The following options may be used to create free space during load:

- **PCTFREE** can be set for indexes and tables; the default value is 10% free for indexes, 0% free for tables. The range of PCTFREE is 0 to 99. Allocating more index free space could reduce page splits which occur when more rows are inserted or loaded into the table. Allocating more table free space could ensure that space is available on the desired page when attempting to insert a row into a table with a clustered index.
- **PAGEFREESPACE=x** is a Load Utility parameter that specifies the percentage of each page that is to be left as free space. The value set by PAGEFREESPACE overrides the PCTFREE value specified for the table.

STATISTICS YES option may be used on the Load Utility to update statistics during execution of the load when the load is run in REPLACE mode.

Overview of the db2move tool

- Moves data between different DB2 databases that may reside on different servers
- Useful when a large number of tables need to be copied from one database to another
- The utility can run in one of three modes:
 - Export — the EXPORT utility is used to export data from the table or tables specified into data files of type IXF
 - Import — the IMPORT utility used to import data files of type IXF into a given database
 - Load — the input files specified in the **db2move.lst** file are loaded into the tables using the LOAD utility

Tables with structured type columns (such as XML) are not moved when this tool is used.

© Copyright IBM Corporation 2007

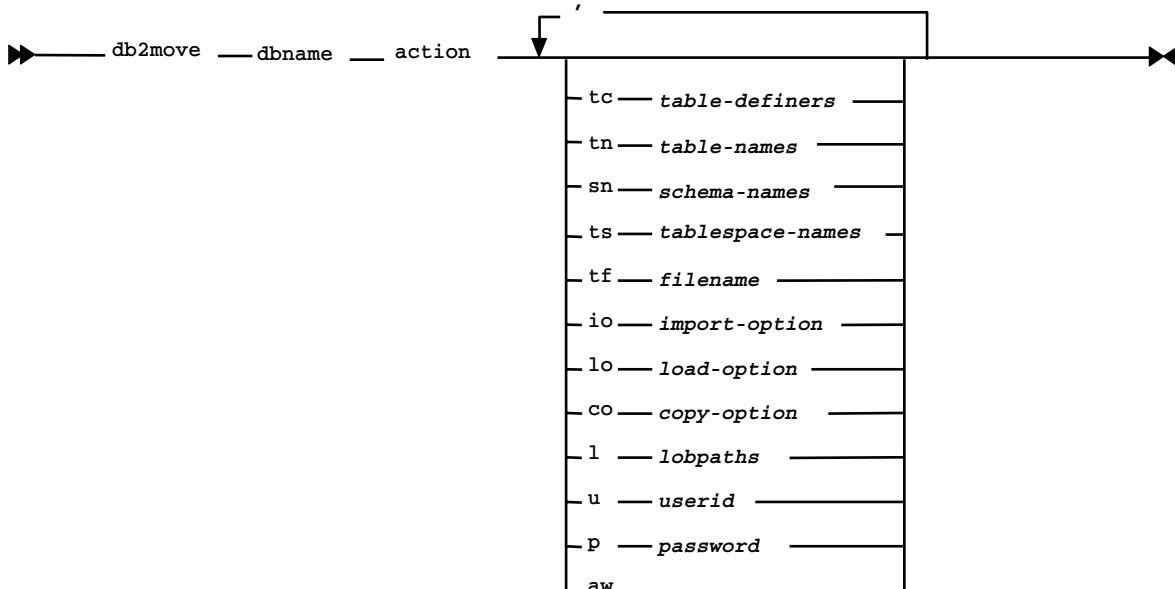
Figure 6-31. Overview of the db2move tool

CF238.3

Notes:

This tool, when used in the EXPORT/IMPORT/LOAD mode, facilitates the movement of large numbers of tables between DB2 databases located on workstations. The tool queries the system catalog tables for a particular database and compiles a list of all user tables. It then exports these tables in PC/IXF format. The PC/IXF files can be imported or loaded to another local DB2 database on the same system, or can be transferred to another workstation platform and imported or loaded to a DB2 database on that platform. Tables with structured type columns are not moved when this tool is used. When used in the COPY mode, this tool facilitates the duplication of a schema.

db2move tool syntax



© Copyright IBM Corporation 2007

Figure 6-32. db2move tool syntax

CF238.3

Notes:

db2move tool command parameters:

dbname Name of the database.

action Must be one of:

EXPORT Exports all tables that meet the filtering criteria in options. If no options are specified, exports all the tables. Internal staging information is stored in the **db2move.lst** file.

IMPORT Imports all tables listed in the internal staging file **db2move.lst**. Use the **-io** option for IMPORT specific actions.

LOAD Loads all tables listed in the internal staging file **db2move.lst**. Use the **-lo** option for LOAD specific actions.

	COPY	Duplicates a schema(s) into a target database. Use the -sn option to specify one or more schemas. See the -co option for COPY specific options. Use the -tn or -tf option to filter tables in LOAD_ONLY mode. See below for a list of files that are generated during each action.
-tc	table-definers. The default is all definers.	This is an EXPORT action only. If specified, only those tables created by the definers listed with this option are exported. If not specified, the default is to use all definers.
-tn	table-names. The default is all user tables.	This is an EXPORT or COPY action only. If specified, only those tables whose names match exactly those in the specified string are exported or copied. If not specified, the default is to use all user tables.
-ts	tablespace-names. The default is all table spaces.	This is an EXPORT action only. If this option is specified, only those tables that reside in the specified table space will be exported. If the asterisk wildcard character (*) is used in the table space name, it will be changed to a percent sign (%) and the table name (with percent sign) will be used in the LIKE predicate in the WHERE clause. If the -ts option is not specified, the default is to use all table spaces.
-tf	filename	This is an EXPORT or COPY action only. If specified, only the tables listed in the given file will be exported or copied. The tables should be listed one per line, and each table should be fully qualified.
-io	import-option. The default is REPLACE_CREATE.	Valid options are: INSERT, INSERT_UPDATE, REPLACE, CREATE, and REPLACE_CREATE.
-lo	load-option. The default is INSERT.	Valid options are: INSERT and REPLACE.
-co	When the db2move action is COPY, the following -co follow-on options will be available:	

```
"TARGET_DB <db name> [USER <userid> USING <password>]"  
"MODE" DDL_AND_LOAD  
LOAD_ONLY  
"SCHEMA_MAP"  
"NONRECOVERABLE"  
"OWNER"  
"TABLESPACE_MAP"
```

- l lobpaths.
For IMPORT and EXPORT, if this option is specified, it will be also used for XML paths. The default is the current directory.
- u userid.
The default is the logged on user ID. Both user ID and password are optional.
- p Password.
The default is the logged on password. Both user ID and password are optional.
- aw Allow Warnings.
When '-aw' is not specified, tables that experience warnings during export are not included in the db2move.lst file (although that table's .ixf file and .msg file are still generated).

db2move tool examples

- To export all tables in the SAMPLE database (using default values for all options), issue:
`db2move sample export`
- To export all tables created by userid1 or user IDs LIKE us%rid2, and with the name tbname1 or table names LIKE %tbname2, issue:
`db2move sample export -tc userid1,us*rid2 -tn
tbname1,*tbname2`
- To import all tables in the SAMPLE database (LOB paths D:\LOBPATH1 and C:\LOBPATH2 are to be searched for LOB files; this example is applicable to Windows operating systems only), issue:
`db2move sample import -l D:\LOBPATH1,C:\LOBPATH2`
- To load all tables in the SAMPLE database (/home/userid/lobpath subdirectory and the tmp subdirectory are to be searched for LOB files; this example is applicable to Linux and UNIX-based systems only), issue:
`db2move sample load -l /home/userid/lobpath,/tmp`

© Copyright IBM Corporation 2007

Figure 6-33. db2move tool examples

CF238.3

Notes:

Usage note:

Loading data into tables containing XML columns is not supported. The workaround is to manually issue the IMPORT or EXPORT commands, or use the db2move -Export and db2move -Import behavior. If these tables also contain generated always identity columns, data cannot be imported into the tables.

Overview of the db2look tool

- Generates a report on the statistics in the database
- Extracts the DDL statements for the creation of tables, indexes, views, and so on, that are needed to generate a CLP script file used to recreate database objects
- Extracts statistics that are stored in a script
 - Can be used to update the statistics
 - Model a development database to simulate a production database by updating the catalog statistics

© Copyright IBM Corporation 2007

Figure 6-34. Overview of the db2look tool

CF238.3

Notes:

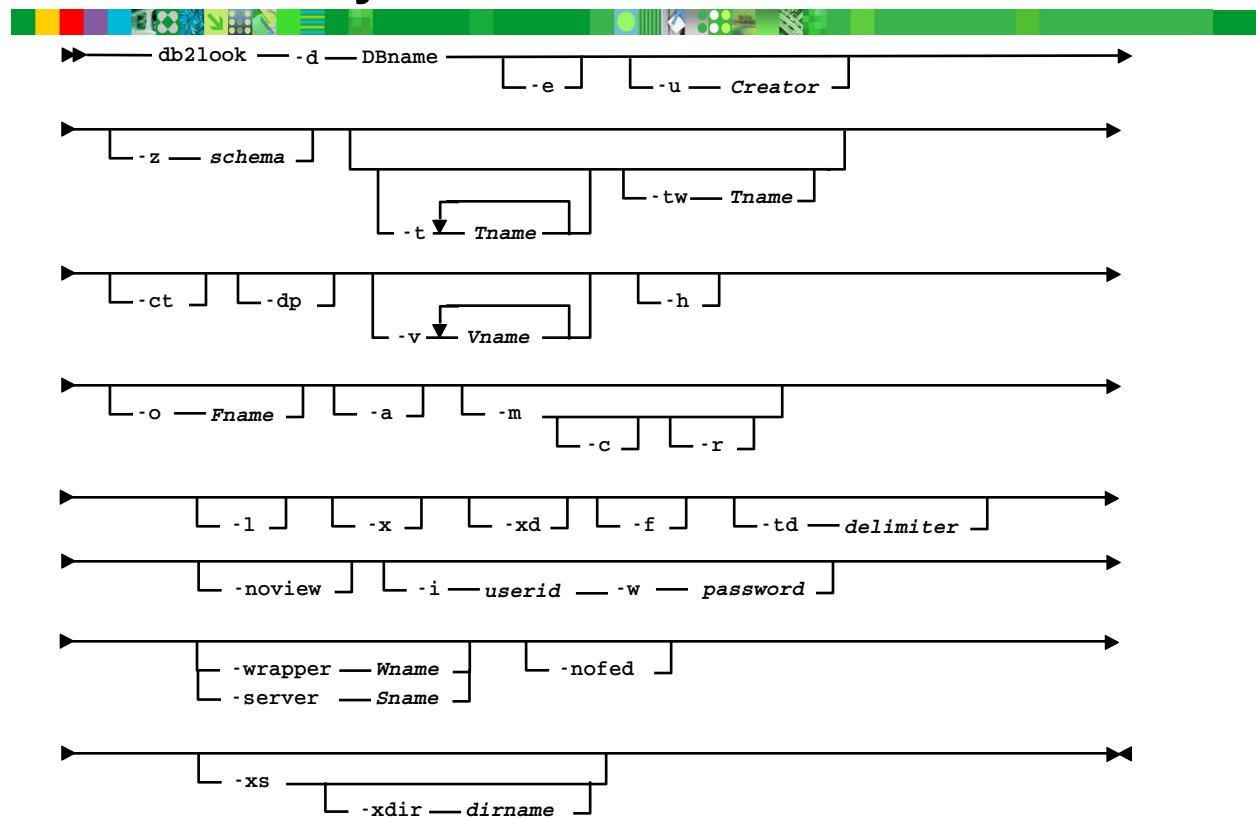
This tool extracts the required Data Definition Language (DDL) statements to reproduce the database objects of a production database on a test database. The db2look command generates the DDL statements by object type.

This tool can generate the required UPDATE statements used to replicate the statistics on the objects in a test database. It can also be used to generate the UPDATE DATABASE CONFIGURATION and UPDATE DATABASE MANAGER CONFIGURATION commands and the db2set commands so that query optimizer-related configuration parameters and registry variables on the test database match those of the production database.

It is often advantageous to have a test system contain a subset of the production system's data. However, access plans selected for such a test system are not necessarily the same as those that would be selected for the production system. Both the catalog statistics and the configuration parameters for the test system must be updated to match those of the production system. Using this tool makes it possible to create a test database where access plans are similar to those that would be used on the production system.

You should check the DDL statements generated by the db2look command since they might not exactly reproduce all characteristics of the original SQL objects. For table spaces on partitioned database environments, DDL might not be complete if some database partitions are not active. Make sure all database partitions are active using the ACTIVATE command.

db2look tool syntax



© Copyright IBM Corporation 2007

Figure 6-35. db2look tool syntax

CF238.3

Notes:

db2look command parameters:

- d** DBname
Alias name of the production database that is to be queried. DBname can be the name of a DB2 Database for Linux, UNIX, and Windows or DB2 Version 9.1 for z/OS (DB2 for z/OS) database.
- e** Extract DDL statements for database objects.
- u** Creator - Creator ID.
Limits output to objects with this creator ID.
- z** schema -Schema name.
Limits output to objects with this schema name.
- t** Tname1 Tname2... TnameN - Table name list.
Limits the output to particular tables in the table list. The maximum number of tables is 30. Table names are separated by a blank space.

- tw Tname
Generates DDL for table names that match the pattern criteria specified by Tname. Also generates the DDL for all dependent objects of all returned tables.
- ct
Generate DDL by object creation time. Generating DDL by object creation time will not guarantee that all the object DDLs will be displayed in correct dependency order.
- dp
Generate DROP statement before CREATE statement. The DROP statement might not work if there is an object that depends on the dropped object.
- v Vname1 Vname2... VnameN
Generates DDL for the specified views. The maximum number of views is 30.
- h
Display help information.
- o Fname
If using LaTeX format, write the output to filename.tex. If using plain text format, write the output to filename.txt. Otherwise, write the output to filename.sql.
- a
When this option is specified the output is not limited to the objects created under a particular creator ID. All objects, including inoperative objects, created by all users are considered.
- m
Generates the required UPDATE statements to replicate the statistics on tables, statistical views, columns and indexes.
 - c When this option is specified in conjunction with the -m option, the db2look command does not generate COMMIT, CONNECT and CONNECT RESET statements. The default action is to generate these statements.
 - r When this option is specified in conjunction with the -m option, the db2look command does not generate the RUNSTATS command. The default action is to generate the RUNSTATS command.
- l
If this option is specified, then the db2look command will generate DDL for user defined table spaces, database partition groups and buffer pools.
- x
If this option is specified, the db2look command will generate authorization DDL (GRANT statement, for example).
- xd
If this option is specified, the db2look command will generate all authorization DDL including authorization DDL for objects whose authorizations were granted by SYSIBM at object creation time.

-f	Use this option to extract the configuration parameters and registry variables that affect the query optimizer.
-td	delimiter Specifies the statement delimiter for SQL statements generated by the db2look command. If this option is not specified, the default is the semicolon (;).
-noview	If this option is specified, CREATE VIEW DDL statements will not be extracted.
-i	userid Use this option when working with a remote database.
-w	password Used with the -i option, this parameter allows the user to run the db2look command against a database that resides on a remote system.
-wrapper	Wname Generates DDL statements for federated objects that apply to this wrapper.
-server	Sname Generates DDL statements for federated objects that apply to this server.
-nofed	Specifies that no federated DDL statements will be generated.
-xs	Exports all files necessary to register XML schemas and DTDs at the target database, and generates appropriate commands for registering them.
-xdir	dirname Places exported XML-related files into the given path. If this option is not specified, all XML-related files will be exported into the current directory.

db2look tool examples

- Generate the DDL statements for objects created by user *walid* in database DEPARTMENT. The **db2look** output is sent to file db2look.sql:
`db2look -d department -u walid -e -o db2look.sql`
- Generate the DDL statements for objects that have schema name *ianhe*, created by user *walid*, in database DEPARTMENT. The **db2look** output is sent to file db2look.sql:
`db2look -d department -u walid -z ianhe -e -o db2look.sql`
- Generate the UPDATE statements to replicate the statistics for the database objects created by user *walid* in database DEPARTMENT. The output is sent to file db2look.sql:
`db2look -d department -u walid -m -o db2look.sql`
- Generate both the DDL statements for the objects created by user *walid* and the UPDATE statements to replicate the statistics on the database objects created by the same user. The **db2look** output is sent to file db2look.sql:
`db2look -d department -u walid -e -m -o db2look.sql`

© Copyright IBM Corporation 2007

Figure 6-36. db2look tool examples

CF238.3

Notes:

Usage note:

On Windows operating systems, the db2look command must be run from a DB2 command window.

More examples:

Generate the DDL statements for all objects (federated and non-federated) in the federated database FEDDEPART. For federated DDL statements, only those that apply to the specified wrapper, FEDWRAP, are generated. The db2look output is sent to standard output:

```
db2look -d feddepart -e -wrapper fedwrap
```

Generate a script file that includes only non-federated DDL statements. The following system command can be run against a federated database (FEDDEPART) and yet only produce output like that found when run against a database which is not federated. The db2look output is sent to a file out.sql:

```
db2look -d feddepart -e -nofed -o out
```

Generate the DDL statements for objects that have schema name walid in the database DEPARTMENT. The files required to register any included XML schemas and DTDs are exported to the current directory. The db2look output is sent to file db2look.sql:

```
db2look -d department -z walid -e -xs -o db2look.sql
```

Generate the DDL statements for objects created by all users in the database DEPARTMENT. The files required to register any included XML schemas and DTDs are exported to directory /home/ofer/ofer/. The db2look output is sent to standard output:

```
db2look -d department -a -e -xs -xdir /home/ofer/ofer/
```

Example output from db2look:

```
db2look -d sample -z -e db2user1 -t department -o sample.tex

-----
-- DDL Statements for table "DB2USER1"."DEPARTMENT"
-----

CREATE TABLE "DB2USER1"."DEPARTMENT"  (
  "DEPTNO" CHAR(3) NOT NULL ,
  "DEPTNAME" VARCHAR(36) NOT NULL ,
  "MGRNO" CHAR(6) ,
  "ADMRDEPT" CHAR(3) NOT NULL ,
  "LOCATION" CHAR(16) )
IN "USERSPACE1" ;

-- DDL Statements for indexes on Table "DB2USER1"."DEPARTMENT"

CREATE INDEX "DB2USER1"."XDEPT2" ON "DB2USER1"."DEPARTMENT"
("MGRNO" ASC)
ALLOW REVERSE SCANS;

-- DDL Statements for indexes on Table "DB2USER1"."DEPARTMENT"

CREATE INDEX "DB2USER1"."XDEPT3" ON "DB2USER1"."DEPARTMENT"
("ADMRDEPT" ASC)
ALLOW REVERSE SCANS;

-- DDL Statements for primary key on Table "DB2USER1"."DEPARTMENT"

ALTER TABLE "DB2USER1"."DEPARTMENT"
ADD CONSTRAINT "PK_DEPARTMENT" PRIMARY KEY
("DEPTNO") ;

-- DDL Statements for aliases based on Table "DB2USER1"."DEPARTMENT"

CREATE ALIAS "DB2USER1"."DEPT" FOR "DB2USER1"."DEPARTMENT";
```

Unit summary

Having completed this unit, you should be able to:

- Review the INSERT statement and recognize its limitations
- Review differences between IMPORT and LOAD
- Learn the EXPORT, IMPORT, and LOAD syntax
- Explore the use of Exception Tables and Dump-Files
- Distinguish the Table States:
 - Load pending, Set Integrity Pending
- Learn why and how to use the SET INTEGRITY command
- Explore use of the **db2move** and **db2look** commands

© Copyright IBM Corporation 2007

Figure 6-37. Unit summary

CF238.3

Notes:

Unit 7. Backup and recovery

What this unit is about

This unit provides you with the information necessary to develop a recovery strategy to support your installation's business requirements. You can use the information in this unit, as well as the product documentation, to plan, define, and implement a recovery strategy.

What you should be able to do

After completing this unit, you should be able to:

- Describe the major principles and methods for backup and recovery
- State the three types of recovery used by DB2
- Explain the importance of logging for backup and recovery
- Describe how data logging takes place, including circular logging and archival logging
- Use the BACKUP, RESTORE, and ROLLFORWARD commands
- Perform a table space backup and recovery
- Restore a database to the end of logs or to a point-in-time
- Discuss the configuration parameters and the recovery history file and use these to handle various backup and recovery scenarios

How you will check your progress

Accountability:

- Checkpoint
- Lab exercises on backup and recovery

References

SC09-4831-00 *Data Recovery and High Availability Guide and Reference*

Unit objectives



After completing this unit, you should be able to:

- Describe the major principles and methods for backup and recovery
- State the three types of recovery used by DB2
- Explain the importance of logging for backup and recovery
- Describe how data logging takes place, including circular logging and archival logging
- Use the BACKUP, RESTORE, and ROLLFORWARD commands
- Perform a table space backup and recovery
- Restore a database to the end of logs or to a point-in-time
- Discuss the configuration parameters and the recovery history file and use these to handle various backup and recovery scenarios

© Copyright IBM Corporation 2007

Figure 7-1. Unit objectives

CF238.3

Notes:

7.1 Introduction to backup and recovery

Backup and recovery

- Backup is *Job #1* for the system administrator — without it you cannot guarantee that your data will be available in the future
- *Backup is for Recovery*
- The system administrator is ultimately responsible for:
 - Establishing a backup policy
 - Enforcing the policy and verifying that the policy has been carried out correctly
 - Testing recovery using the products of backup
- Backup involves both the database and any support files that are not part of the database itself:
 - Database objects
 - Table spaces
 - Log files
 - Some non-database files

© Copyright IBM Corporation 2007

Figure 7-2. Backup and recovery

CF238.3

Notes:

Backup is done for precisely one purpose: database recovery *when* it is needed.

You should note the use of the word *when* rather than *if*. We will see that recovery is needed for normal functioning of the system (*crash recovery*) and for abnormal functioning of the database system (*version recovery/restore* and *rollforward*).

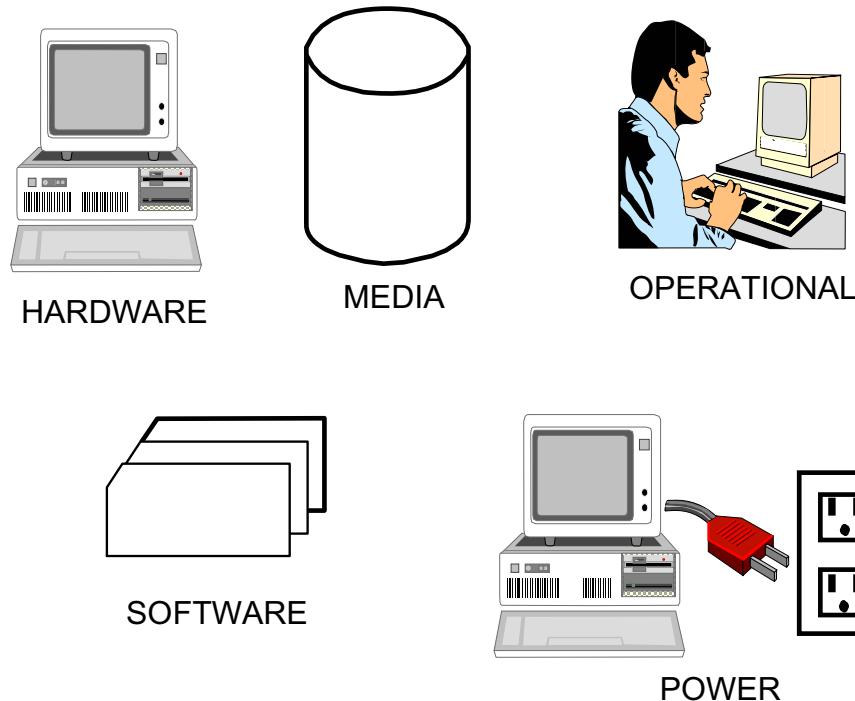
You should also note that you as system administrator are the primary focus of the recovery process. Hence you cannot hide behind someone else not doing their job correctly. You are ultimately responsible for establishing, enforcing, and testing any recovery policy and process. You need to review the steps that will be undertaken by others to ensure that they are carried out as laid down in policy so that when the time comes you will have the files (generally tapes) needed to perform recovery.

A company that loses its data is seriously in danger of failing.

Some data that must be backed up is not backed up as part of the database backup. You need to become aware of the complete picture and take responsibility for all of this.

And, it *is* your Job #1.

Types of failure



© Copyright IBM Corporation 2007

Figure 7-3. Types of failure

CF238.3

Notes:

DB2 provides for recovering from a variety of data processing problems.

The degree of recovery depends 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 listed, the database administrator needs to take full advantage of all the recovery features of DB2. However:

- An installation must define its recovery needs.

Those needs may not require use of all the recovery features available.

Database objects

Database

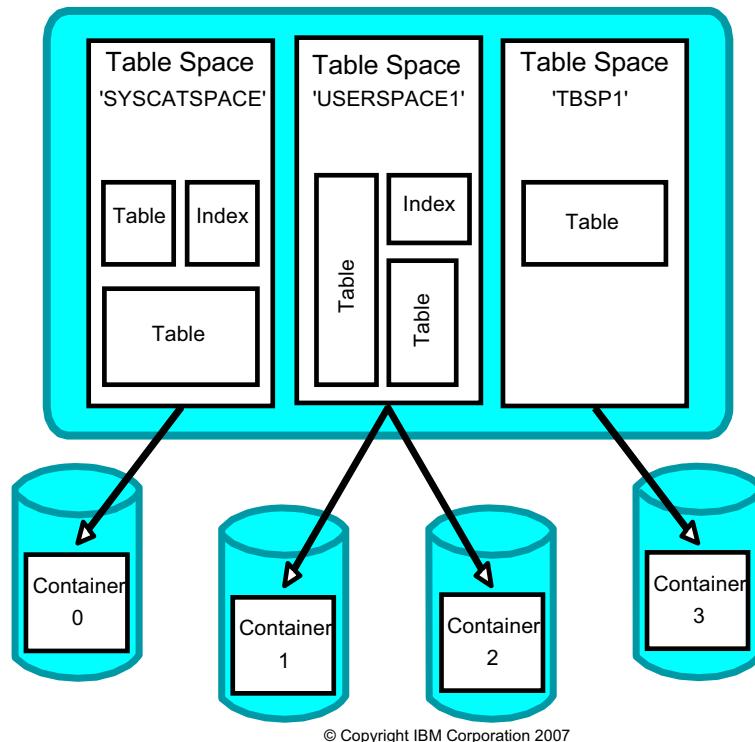


Figure 7-4. Database objects

CF238.3

Notes:

The data in a database is placed into table spaces.

A table space is a storage model that provides a level of indirection between a database and the tables stored within that database.

Table spaces allow you to assign the location of database and table data directly onto containers. (A container can be a directory name, a device name, or a file name.)

A single table space can span several containers. For performance reasons, each container should use a different disk.

The architecture shown in the diagram has impacts on recovery processes. Two of these impacts are:

- The data for different tables might share the same table space. A restore and roll-forward recovery of the table space will restore and recover *all* tables in that table space.

- While it is a good practice to let related tables share a table space, you should consider storing unrelated tables separately in order to minimize recovery processing time when restoring data at the table space level.

The DB2 backup utility allows for backing up at the database and table space levels. Export or *db2move* could be used to back up individual tables.



Note

Dropped Table Recovery

Q: *How do I ensure that critical tables in my DB2 database are recoverable in case I inadvertently drop them?*

We will see later that DB2 allows you to recover a database's data using a database RESTORE operation. You can use the RESTORE operation in conjunction with the ROLLFORWARD operation to restore the database to a point before the table was dropped, but restoring the entire database can be very time consuming. More importantly, during this database-level operation, you cannot access your database, and furthermore there is always the finite chance that your backup media are defective.

There is a *dropped table recovery option* available for table spaces to solve this problem. This option is turned on by default when tables are created using the command line, but it is an option that must be specifically selected if you create a table space with the GUI.

To make sure a given name space has table recovery enabled, query the DROP_RECOVERY column of the SYSCAT.TABLESPACES catalog table:

```
db2 => SELECT tbspace, drop_recovery FROM syscat.tablespaces
```

To enable the DROPPED TABLE RECOVERY option after you have created a table space, use:

```
ALTER TABLESPACE mytablespace DROPPED TABLE RECOVERY on
```

If you drop a table, and don't remember its name, you can use:

```
db2 => LIST HISTORY DROPPED TABLE ALL FOR mydatabase
```

We do not have space here to describe the whole process for RECOVER and ROLLFORWARD needed to have restore this table. This type of recovery and rollforward is covered in *CF49 DB2 Advanced Recovery Workshop*.

Obviously you need to think ahead. It is too late to turn on DROPPED TABLE RECOVER after the table has already been dropped.

Forms of backup

- Table space backup
 - Full / incremental backup
 - Online / offline backup
- Log backup
 - Circular / archival logging
 - Single / mirror logging
- Alternate disaster and recovery approaches
 - High availability disaster recovery (HADR)
 - Data replication
- Other issues — towards a complete backup and recovery plan
 - Disk / tape / transmission to a remote site — offsite storage
 - Storage management systems (TSM, XBSA, ...)
 - Other critical files and hardware requirements
 - Backup and recovery sites
 - Network considerations

© Copyright IBM Corporation 2007

Figure 7-5. Forms of backup

CF238.3

Notes:

Backup is not monolithic.

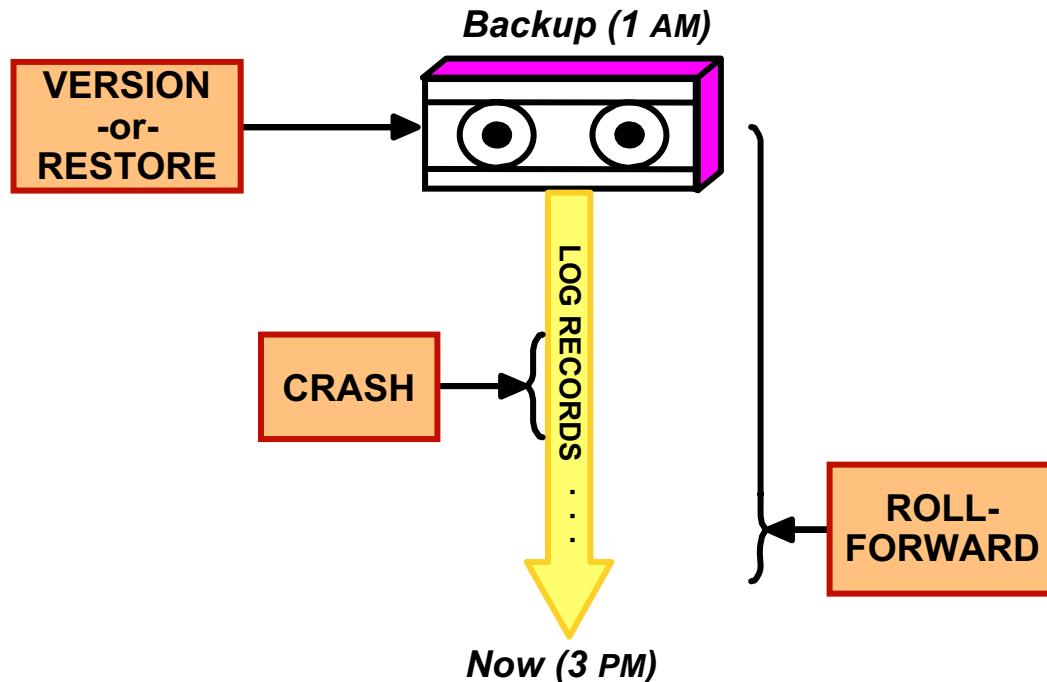
We need to learn about the types of table space backup and the types of log backup.

We need to understand the place and role of alternative and complimentary disaster and recovery approaches (but they are not discussed in this course).

And, finally, we need to understand the related technology, including the role of storage management systems such as Tivoli Storage Manager (TSM) or other Backup Services APIs (XBSA) and work towards a complete backup and recovery plan.

7.2 Methods of recovery and logging

Methods of recovery



© Copyright IBM Corporation 2007

Figure 7-6. Methods of recovery

CF238.3

Notes:

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, power interruptions, and application failures. You need to know that you can back up your database, or individual table spaces, and then rebuild them should they be damaged or corrupted. The rebuilding of these objects is called recovery. There are three types or methods of recovery:

1. Crash
2. Version (or Restore)
3. Roll Forward

Crash Recovery

Crash Recovery uses the logs to recover from power interrupts or application abends. This type of recovery is normally automated by using the default value for the configuration parameter **autorestart**. If autorestart is not used, manual intervention would be necessary to restart a database requiring crash recovery.

Version or Restore Recovery

Version or Restore Recovery uses a backup image 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.

Roll Forward Recovery

Roll Forward Recovery uses a backup copy of a database or table spaces to replace the current data and then applies log records to recover changes made after the backup image was created. This type provides recovery from media, hardware, operational, and software problems, and the recovery can be to a point in time or to the last committed unit of work. Roll forward recovery using a recent backup will require fewer logs to be applied than a roll forward using an older backup, so the frequency of backup will still be a consideration for the database administrator.

Introduction to logging

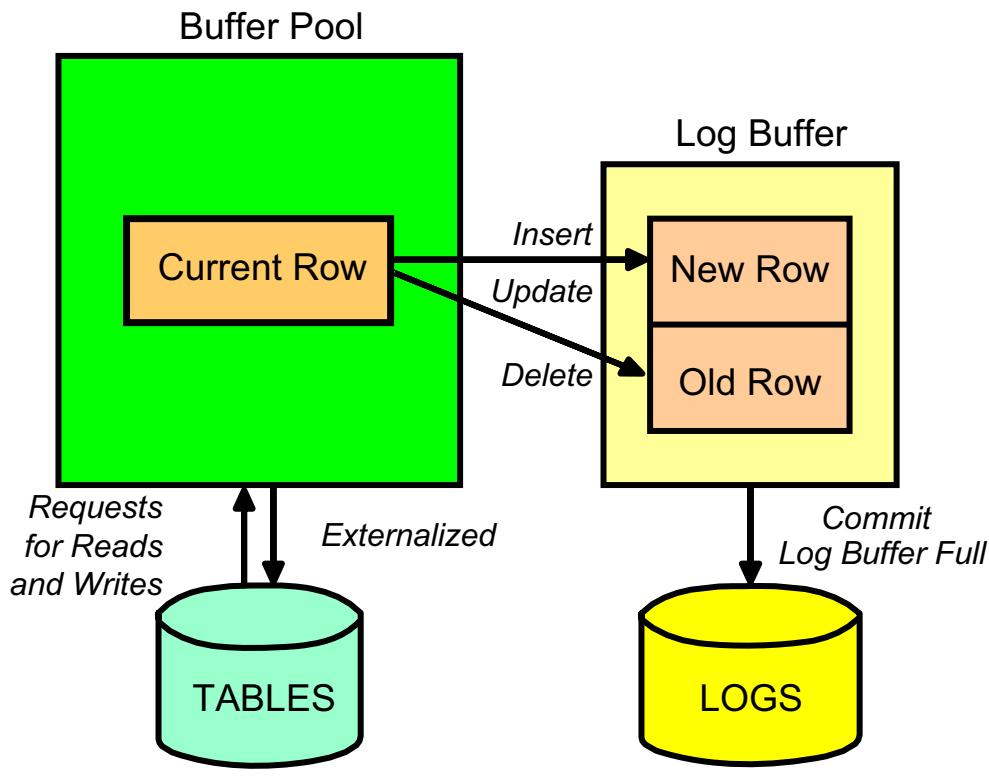


Figure 7-7. Introduction to logging

CF238.3

Notes:

Transaction logging is the process that records each change to a database to permit recovery.

The database manager maintains a log of recent changes made to a database so that the recovery process can restore the database to a consistent state. The log records are written in a log buffer to reflect the activity occurring against data in the buffer pool. The data that is logged includes all column data on an insert or delete, but only from first-changed-column to last-changed-column on an update.

At commit, the log records MUST be written from the log buffer to the log files on disk in order to guarantee recoverability. The application issuing the commit will not receive confirmation that the commit has successfully completed until the log buffer has been written.

It is possible that log records will be written to disk before a commit, for example when the log buffer fills up. However, this does not impact the integrity of the system, because the execution of a commit itself is logged. A unit of work that has started and is externalized to

the log file is not considered complete until the commit associated with that unit of work is also externalized to the log file.

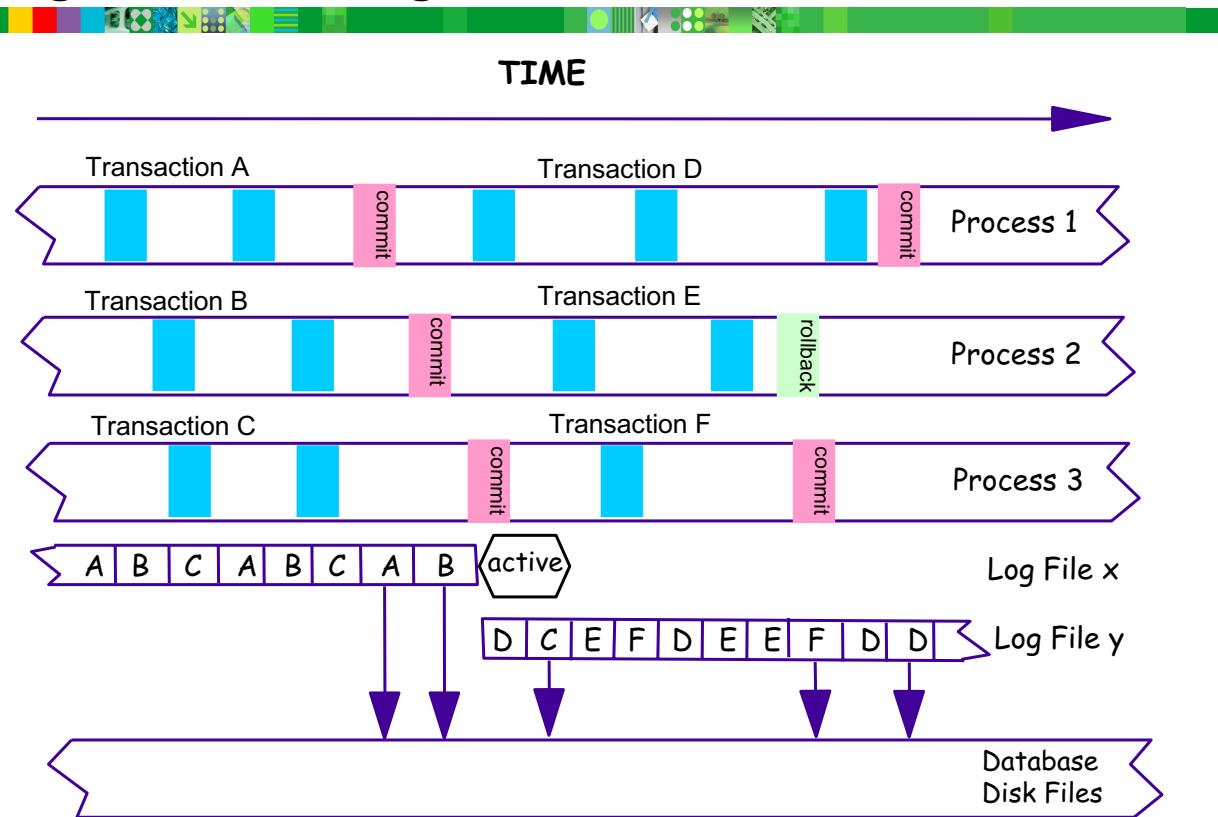
The pages in the buffer pool changed by a unit of work that has been committed do not need to be written to disk at commit. The log records contain the information required to recover such changes in the event of a recovery situation. It is desirable to leave pages in the buffer pool if the data they contain is accessed frequently.

If you are logging large object (LOB) data, you have to consider the impact to 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, your application's performance improves, however its recoverability is sacrificed.

For more information on the merits of logging LOB data, refer to the *Data Recovery and High Availability Guide and Reference*.

DB2 supports two basic types of logging: circular logging and archival logging (log retention logging). Details regarding these two basic types are the subject of subsequent pages.

Log file use during concurrent transactions



© Copyright IBM Corporation 2007

Figure 7-8. Log file use during concurrent transactions

CF238.3

Notes:

The visual shows three transactions initially taking place in parallel: A, B, and C. They happen to be alternating in the changes recorded to the log file — but the actions are independent and hence log file entries can be in any order and with any timing.

Commits take place. In one case there is a rollback.

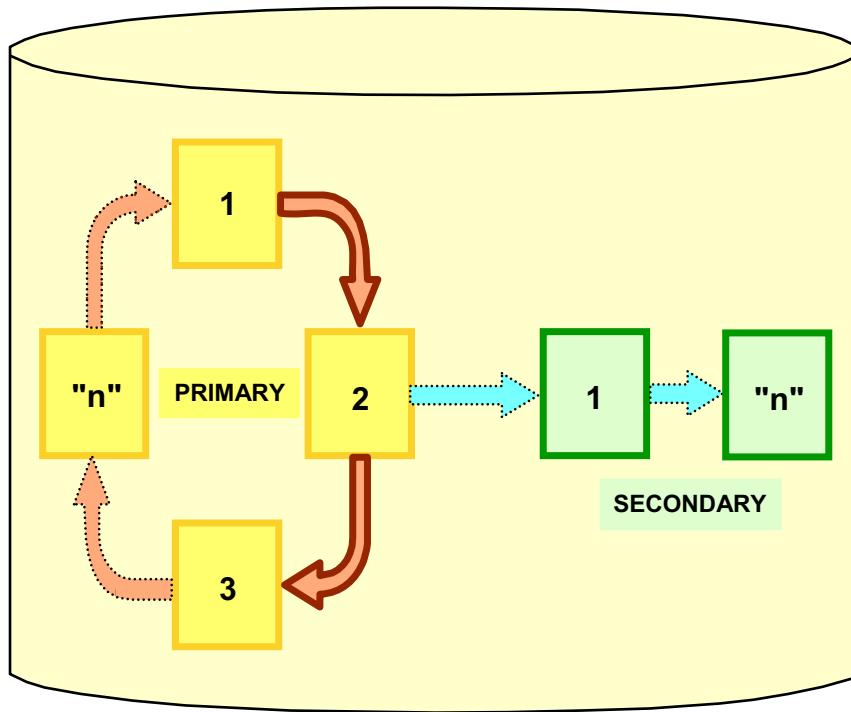
While this is taking place, other transactions also start and complete. These are indicated with other transaction letters (D, E, F, ...) and could occur as part of the work of the same processes or in separate processes.

The log file itself switches from X to Y at an arbitrary point in time as the first file fills.

Also, at any arbitrary point in time (any vertical time slice across all this activity), the system can fail and all information in memory is lost. That which has been completely written to the disk log files is available for crash recovery. Some transactions are complete (commit or rollback) and some transactions are in midstream.

This is the problem that is solved by crash recovery.

Circular logging



© Copyright IBM Corporation 2007

Figure 7-9. Circular logging

CF238.3

Notes:

Circular logging will use the number of primary log files specified via a configuration parameter. The necessary log information is for in-process transactions. The log files are used in sequence. A log file 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.

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.

Primary log files are allocated when the database is created, while secondary log files are allocated as needed. Secondary log files are deallocated once the database manager determines that they are no longer 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

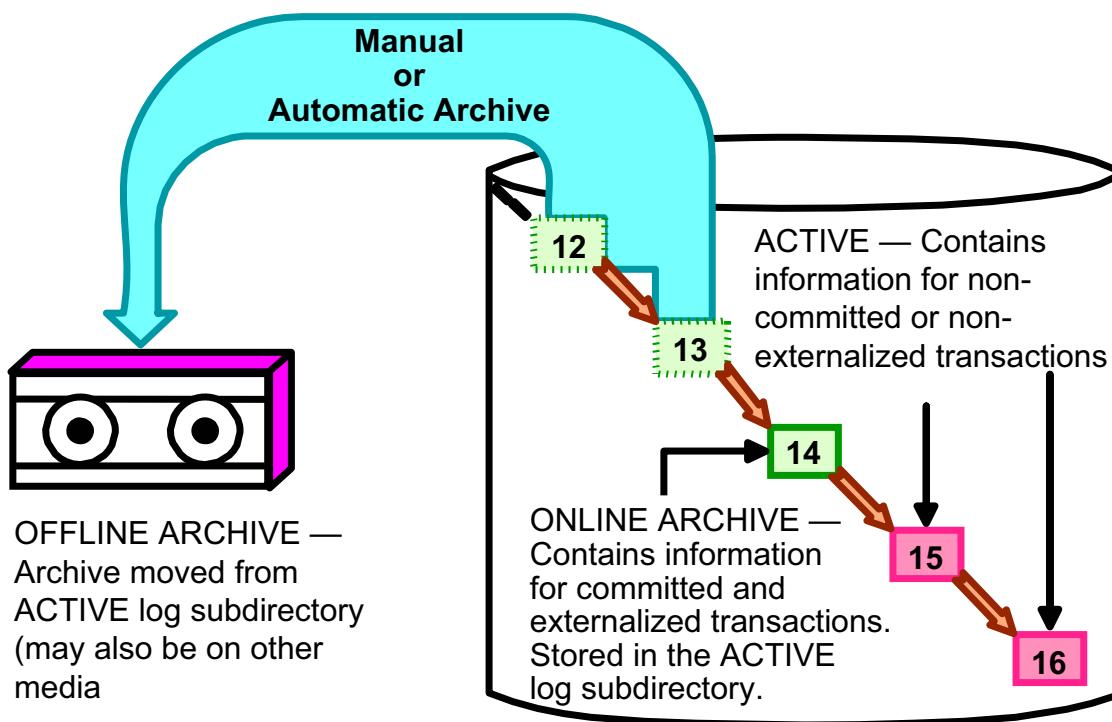
example, submission of an IMPORT utility with a large commit count may require the use of secondary log files. Supporting such applications via primary logs would be wasteful of space, since all primary logs, whether used or not, are allocated when the database is activated.

If DB2 cannot continue logging due to a log full condition, database activity will halt until a log file becomes available.

The logpath should be able to contain the sum of the primary and secondary logs. LOGPRIMARY + LOGSECOND must be less than or equal to 256. LOGFILSIZ has a maximum of 262,144 4 KB pages. The total active log file size limit is 256 GB.

Circular logging provides support for crash and version/restore recovery, but does NOT support roll forward recovery.

Archival logging



© Copyright IBM Corporation 2007

Figure 7-10. Archival logging

CF238.3

Notes:

The second type of logging supported by DB2 is archival logging (log retention logging), where log files are not reused.

When a log becomes full, another log file is allocated. Usually, the database administrator will configure 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.) The number of log files allocated when the database is created is specified by the number of primary logs.

This type of logging is enabled through configuration parameters highlighted later in this unit.

If DB2 allocates the sum of primary and secondary logs as defined in the database configuration file and requires additional space for logging, a log full condition will result. This situation can be caused by an application that attempts to process too large a unit of work. It can also be caused by a configuration that allocates too few log files, or log files that are too small to handle the work load.

The logpath (or newlogpath) should be able to contain two times the sum of the primary and secondary logs plus 1. The maximum total log file size limit is 256 GB (that is, the number of log files (LOGPRIMARY + LOGSECOND) multiplied by the size of each logfile in bytes (LOGFILSIZ * 4096) must be less than 256 GB). LOGPRIMARY + LOGSECOND must be less than or equal to 256. LOGFILSIZ has a maximum of 262,144 4 KB pages.

There are three types of log files associated with log retention logging:

1. Active — 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.)
The active log supports crash recovery.
2. Online Archive — these files contain information related to completed transactions that no longer require crash recovery protection. They are termed *online* because they reside in the same subdirectory as the active log files.
3. Offline archive — 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 user exit.



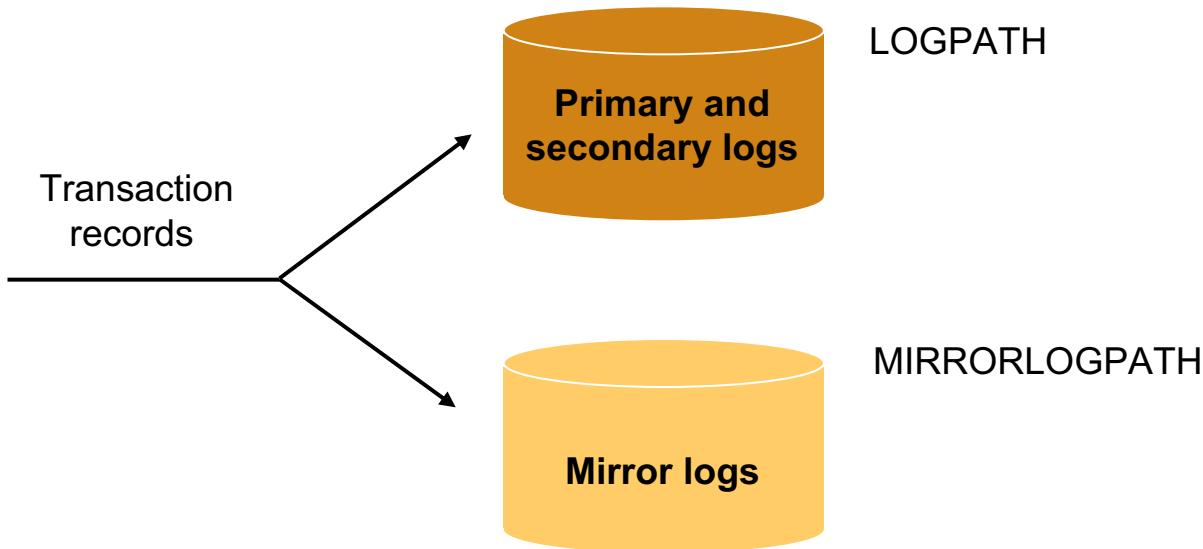
Note

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 and the database deactivates. 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 reallocate primary log files when a new application connects.

The database administrator should consider using the ACTIVATE DATABASE command. This command will keep the database active and prevent log file truncation. However, the administrator must remain aware of the impact on recovery. If the database is truly not being used for an extended period of time, preventing log file truncation will also prevent the most recent log information from being archived. This will make the recovery point for the database to be less than the most recent unit of work if a failure on the log disk occurs. The DEACTIVATE DATABASE command can be specified to allow log file truncation to occur for databases on which the ACTIVATE command has been used.

An additional benefit of the ACTIVATE DATABASE command would be that memory is normally allocated at first connect and released when the database deactivates. This normally occurs when the last application disconnects are retained. This can enhance the performance of databases with periods of inactivity.

Mirrored logging



© Copyright IBM Corporation 2007

Figure 7-11. Mirrored logging

CF238.3

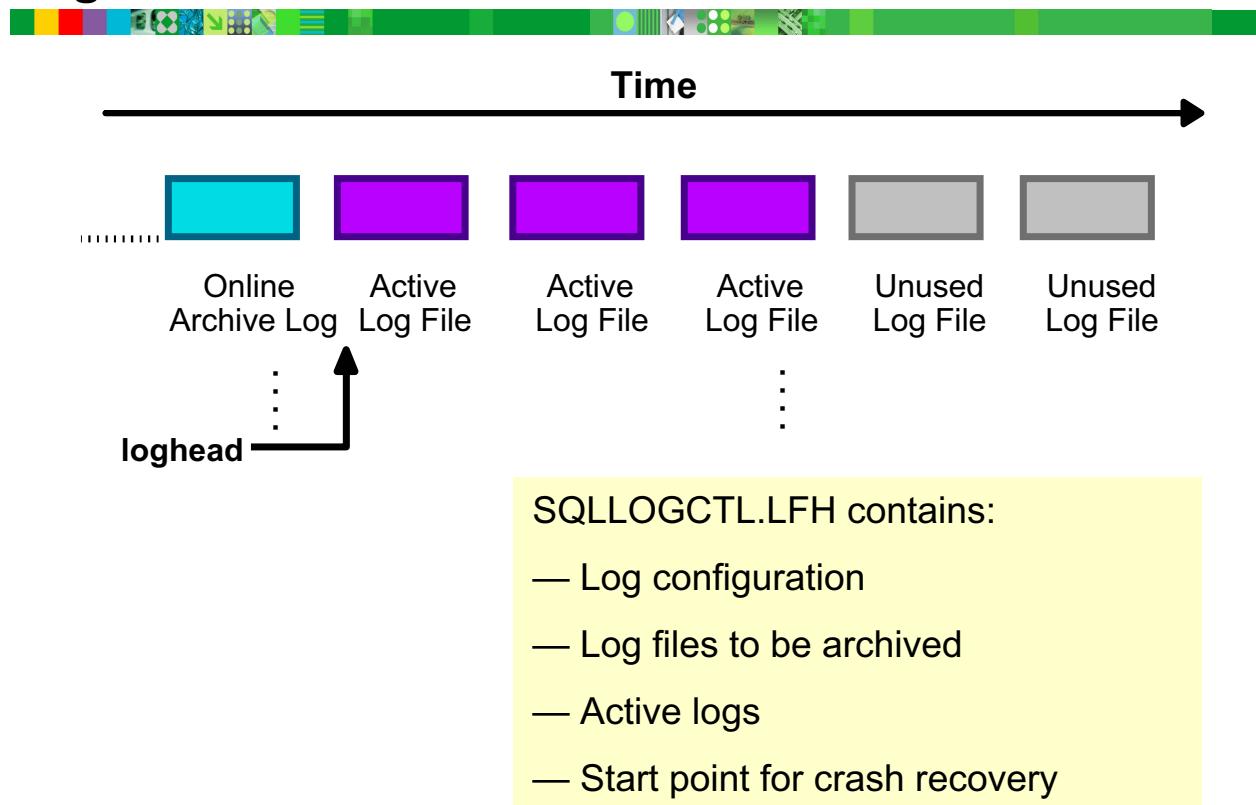
Notes:

Dual or mirrored logging provides a way to maintain mirror copies of both primary and secondary logs — this capability was added to DB2 UDB in v8.1. If either the primary or the secondary log becomes corrupt (but not both), or if the device where either logs is stored becomes unavailable, the database can still be accessed.

Mirrored logs should be kept on different disk drives and preferably on different disk controllers (and, certainly, different RAID controllers). Since log files are written serially, dedicated drives should be preferred as other disk action on the same drive can reduce performance.

Mirrored logging is enabled by setting the MIRRORLOGPATH database configuration parameter to a path where the mirror logs are to be located.

Log file information



© Copyright IBM Corporation 2007

Figure 7-12. Log file information

CF238.3

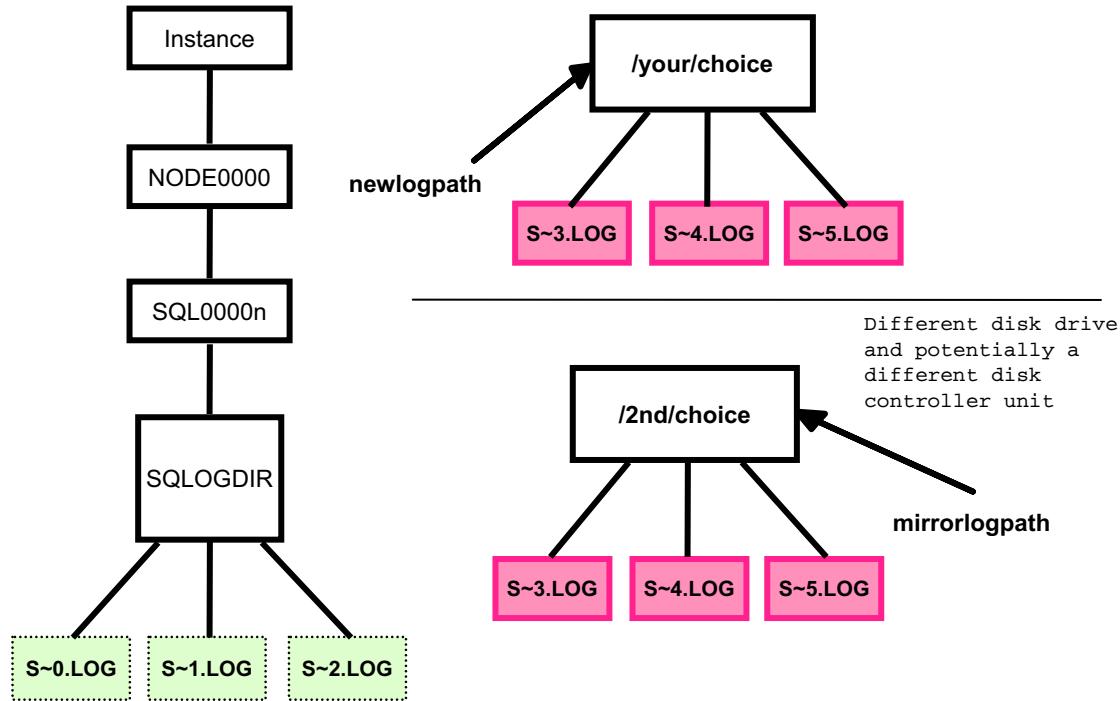
Notes:

DB2 uses a control file to determine the status of log files. The control file identifies the active log file with the *lowest* name. This is called the loghead.

In archival logging, the values for loghead can be valuable to the database administrator. It can be obtained via a GET DATABASE CONFIGURATION command. The value of *First active log file* identifies the loghead.

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.

Location of log files



© Copyright IBM Corporation 2007

Figure 7-13. Location of log files

CF238.3

Notes:

By default, log files are located in SQLOGDIR, which is a subdirectory of the database directory. 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.

The location of log files currently in use is identified by the informational configuration parameter *Path to log files*.

The NEWLOGPATH database configuration parameter allows the administrator to redirect logging support to a specified path. The new path does not become active until the database becomes inactive 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 informational database configuration parameter *Database is consistent* contains this status.) When the database becomes active again, the value in NEWLOGPATH will be used to identify the new location of the log files.

The situation illustrated is not generally desirable. In the illustration, 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 configuration file to indicate a NEWLOGPATH of /usr/your/choice. Assuming all applications completed units of work and disconnected from the database before allocation of **S0000003.LOG**, the logpath was changed to /usr/your/choice at the time a new application connected to the database. The file **S0000003.LOG** was allocated in the new location. This scenario can cause subsequent recovery processes to be more complex than necessary. If a log path change is desired, change the log path **BEFORE** using the database in order to direct the logs to a device that does not contain database files.

If a change to the log path is required after a database has been used, create a database backup after changing the log path.

A recovery strategy involving a change to log path during the recovery process is not recommended.

DB2 supports log mirroring at the database level. Mirroring log files helps protect a database from:

- Accidental deletion of an active log
- Data corruption caused by hardware failure

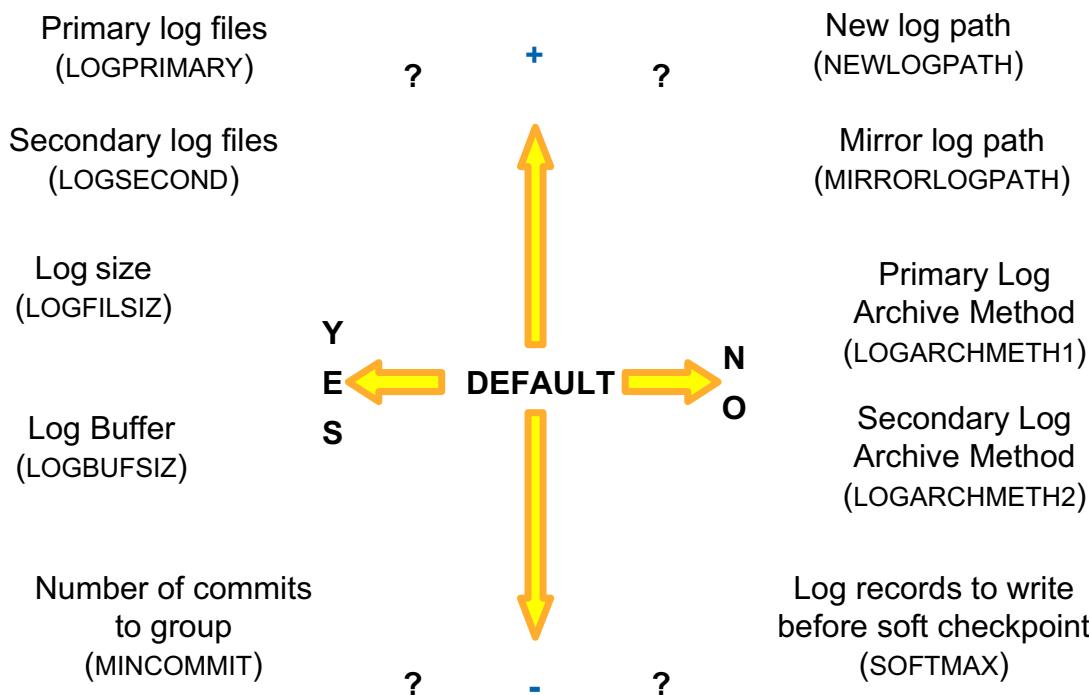
If you are concerned that your active logs may be damaged (as a result of a disk crash), you should consider using the DB2 configuration parameter, MIRRORLOGPATH, to specify a secondary path for the database to manage copies of the active log, mirroring the volumes on which the logs are stored.

The MIRRORLOGPATH configuration parameter allows the database to write an identical second copy of log files to a different path. It is recommended that you place the secondary log path on a physically separate disk (preferably one that is also on a different disk controller). That way, the disk controller cannot be a single point of failure.

When MIRRORLOGPATH is first enabled, it will not actually be used until the next database startup. This is similar to the NEWLOGPATH configuration parameter.

If there is an error writing to either the active log path or the mirror log path, the database will mark the failing path as *bad*, write a message to the administration notification log, and write subsequent log records to the remaining *good* log path only. DB2 will not attempt to use the *bad* path again until the current log file is completed. When DB2 needs to open the next log file, it will verify that this path is valid, and if so, will begin to use it. If not, DB2 will not attempt to use the path again until the next log file is accessed for the first time. There is no attempt to synchronize the log paths, but DB2 keeps information about access errors that occur, so that the correct paths are used when log files are archived. If a failure occurs while writing to the remaining *good* path, the database shuts down.

Configure database logs — parameters



© Copyright IBM Corporation 2007

Figure 7-14. Configure database logs — parameters

CF238.3

Notes:

The configuration parameters listed above can impact logging for the database. The charts on the following pages summarize some considerations regarding these parameters. The proper values for these parameters are highly dependent on the installation requirements, so care must be taken regarding rules of thumb. Consult the *Data Recovery and High Availability Guide and Reference* for further details.

Parameter (scope)	Description	Default (range)	General recommendations
logprimary (db)	Number of primary log files	3 (2-256)	This value represents the number of primary log files that will be allocated to support database logging. 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 logging with highly active systems, it may be necessary to use a value greater than the default so that waits for log allocations are not experienced. In the opposite direction, a database that is not accessed frequently may be properly supported by just two log files, with a savings in disk space.
logsecond (db)	Number of secondary log files	2 (-1; 0-254)	This parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed). When the primary log files become full, the secondary log files (of size logfilsiz) are allocated one at a time as needed, up to a maximum number as controlled by this parameter. Changes to this parameter are immediately effective.
logfilsiz (db)	Size of primary and secondary log file in pages	UNIX: 1000 Intel: 250 (4 - 262144)	This parameter 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 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.

logbufsz (db)	Number of pages used to buffer log records	8 (32-bit: 4 - 4096, 64-bit: 4 - 65535)	This parameter determines the amount of database shared memory that is utilized for buffering log records before physical writes. A large value can improve logging I/O for active databases, but the cost is memory.
mincommit (db)	Number of commits to group	1 (1 - 25)	This parameter indicates that grouping of commits issued by multiple applications is to be attempted, if the value is set 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 change applications that do not execute concurrently with other change applications. Changes to the value specified for this parameter take effect immediately.
newlogpath (db)	New log path for database logs	Null (any valid 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. Set this parameter to direct log files to a disk that does not have high I/O requirements and does not contain the database itself.

mirrorlog-path (db)	Log path for mirrored logs	Null (any valid path)	This parameter allows you to specify a string of up to 242 bytes for the mirror log path. The string must point to a path name, and it must be a fully qualified path name, not a relative path name. If you configure the mirrorlogpath parameter, you must specify a path name for newlogpath, not a raw device. If mirrorlogpath is configured, DB2 will create active log files in both the log path and the mirror log path. All log data will be written to both paths. The mirror log path has a duplicated set of active log files, such that if there is a disk error or human error that destroys active log files on one of the paths, the database can still function.
logarch-meth1 (db)	Primary log archive method	Off (DISK, TSM, Vendor, userexit, logretain)	This parameter is the primary log archive method configuration parameter and is used to specifies the media type of the primary destination for archived logs. <ul style="list-style-type: none"> - DISK:<path> - TSM:[management class name] - Vendor:<vendor library> - USEREXIT - LOGRETAIN
logarchopt1 (db)	Options for logarch-meth1	Null	This parameter specifies the options field for the primary destination for archived logs (if required). For example, <pre>UPDATE DB CFG FOR mydb USING LOGARCHOPT1 -SERVERNAME=mytsmserver2</pre>
logarch-meth2 (db)	Secondary log archive method	Off (DISK, TSM, Vendor, userexit, logretain)	This parameter specifies the media type of the secondary destination for archived logs. If this path is specified, log files will be archived to both this destination and the destination specified by the <i>logarchmeth1</i> database configuration parameter.
logarchopt2 (db)	Options for logarch-meth2	Null	This parameter specifies the options field for the secondary destination for archived logs (if required).

logretain (db)	Log retention logging to be used	No (No Recovery)	<p>Setting this parameter to Recovery indicates that log retention logging is to be used. The log files become archive log files when full instead of being reused. If an installation wishes to enable roll forward recovery, this parameter (or userexit) must be enabled.</p> <p>Now deprecated. Use instead logarchmeth1 and logarchopt1, and also logarchmeth2 and logarchopt2 if needed.</p>
userexit (db)	User exit enable	No (Yes No)	<p>This parameter indicates whether a user exit should be called when processing archive or retrieval requests for log files. Setting this value to yes enables log retention logging and roll forward 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 exit.</p> <p>Now deprecated. Use instead logarchmeth1 and logarchopt1, and also logarchmeth2 and logarchopt2 if needed.</p>
softmax (db)	Log records to write before soft checkpoint	100 (1 - 100* logprimary)	<p>This parameter will influence the number of logs that need to be recovered following a crash (such as a power failure). For example, if the default value is used, the database manager will try to keep the number of logs that need to be recovered to 1. If you specify 300 as the value of this parameter, the database manager will try to keep the number of log files that need to be recovered to 3. To influence the number of logs required for crash recovery, the database manager uses this parameter to trigger the page cleaners to ensure that pages older than the specified recovery window are already written to disk. This parameter will also determine the frequency of soft checkpoints.</p>

Changes to most log configuration parameters are not immediately effective. For example, enabling log retention logging will not change circular logs allocated until all applications disconnect from the database and the database is deactivated. The next activation of the database will cause new allocation of log files to support the new strategy. Only LOGSECOND and MINCOMMIT will be effective as soon as the change is committed, though other parameters may become effective earlier in later releases of DB2.

There are some more logging configuration parameters. Consult the *Data Recovery and High Availability Guide and Reference* for further details.

Configure database logs — GUI (Control Center)

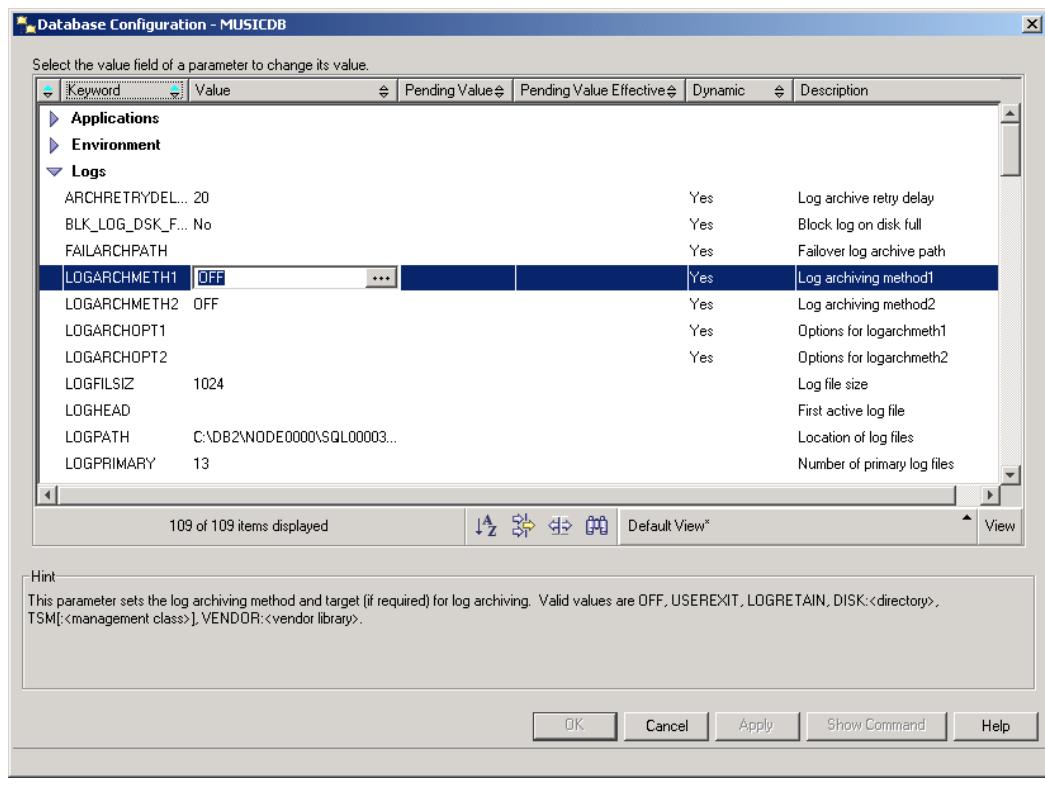


Figure 7-15. Configure database logs — GUI (Control Center)

CF238.3

Notes:

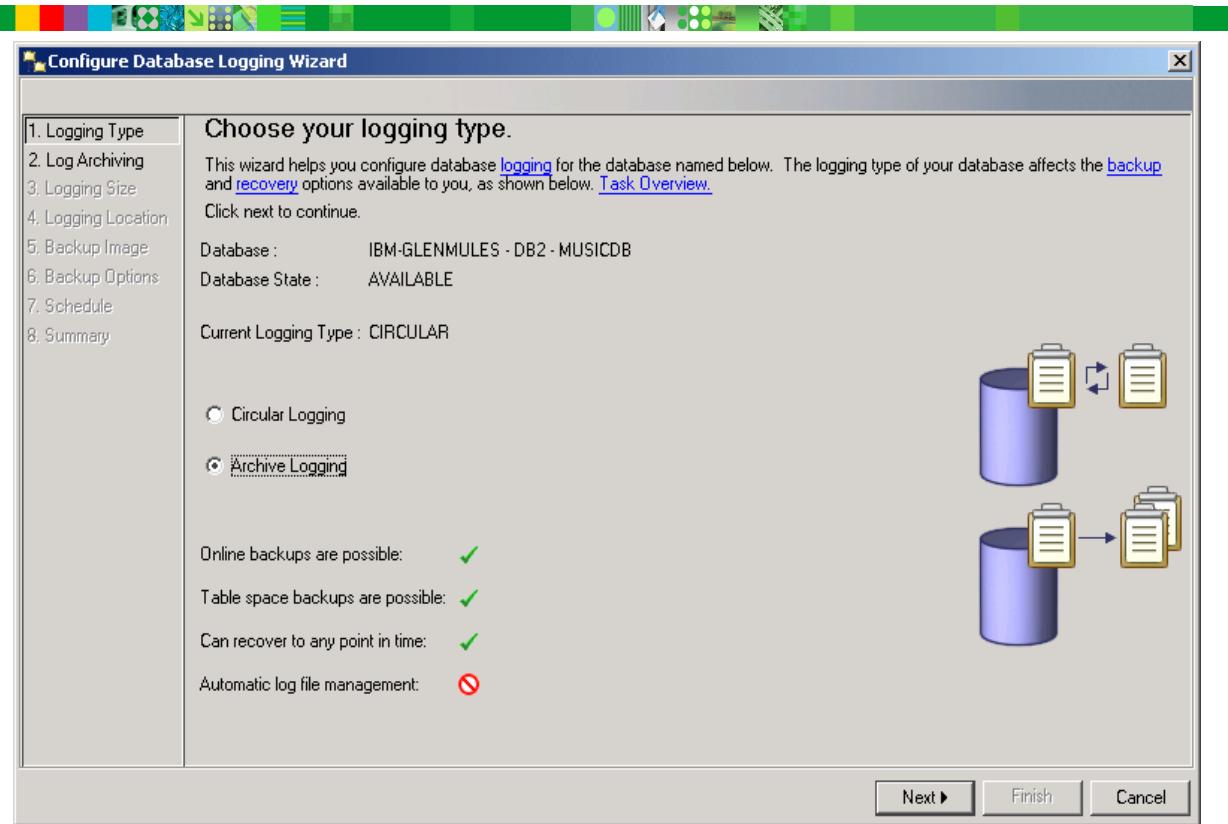
To configure logs through GUI:

- Right-click and select the database
 - Left-click and select **Configure**
 - Scroll down to the parameters affecting logs

Log parameters such as logfilsiz, logprimary, logretain, and many others can be configured from this GUI.

Note that this GUI is used to change any configuration parameters — not just those for logging. The configuration parameters are arranged into about 9 groupings. The ones relevant to logging are gathered under the header logs (the triangle on the left can be flipped to show or hide the configuration parameters in each group).

Configure database logs — Wizard (CC)



© Copyright IBM Corporation 2007

Figure 7-16. Configure database logs — Wizard (CC)

CF238.3

Notes:

To configure logs through the Logging Wizard GUI from the Control Center:

- Right-click and select the database
 - Left-click and select **Configure Database Logging**
 - Select logging type (Circular or Archive)
 - Log Archiving (Manual, User Exit, or DB2)
 - Logging Size
 - Logging Location
 - Run now or schedule for later execution

When you use the *Configure Database Logging* wizard, DB2 performs all the relevant steps for you, such as changing the relevant parameters as well as making a database backup.

Configure database logs — Command line

- Windows: db2 get db cfg for musicdb | find /i "log"
- Linux/UNIX: db2 get db cfg for musicdb | grep -i log
- Update: db2 update db cfg for musicdb using logprimary 15

```

Log retain for recovery status = NO
User exit for logging status = NO
Catalog cache size (4KB) (CATALOGCACHE_SZ) = 260
Log buffer size (4KB) (LOGBUFSZ) = 98
Log file size (4KB) (LOGFILSZ) = 1024
Number of primary log files (LOGPRIMARY) = 13
Number of secondary log files (LOGSECOND) = 4
Changed path to log files (NEWLOGPATH) =
Path to log files = C:\DB2\NODE0000\SQL00003\SQLQDIR\
Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file =
Block log on disk full (BLK_LOG_DSK_FUL) = NO
Percent max primary log space by transaction (MAX_LOG) = 0
Num. of active log files for 1 active UOW (NUM_LOG_SPAN) = 0
Percent log file reclaimed before soft chkpt (SOFTMAX) = 520
Log retain for recovery enabled (LOGRETAIN) = OFF
User exit for logging enabled (USEREXIT) = OFF
HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC
First log archive method (LOGARCHMETH1) = OFF
Options for logarchmeth1 (LOGARCHOPT1) =
Second log archive method (LOGARCHMETH2) = OFF
Options for logarchmeth2 (LOGARCHOPT2) =
Failover log archive path (FAILARCHPATH) =
Number of log archive retries on error (NUMARCHRETRY) = 5
Log archive retry Delay (secs) (ARCHRETRYDELAY) = 20
Log pages during index build (LOGINDEXBUILD) = OFF

```

© Copyright IBM Corporation 2007

Figure 7-17. Configure database logs — Command line

CF238.3

Notes:

The most efficient way of updating configuration parameters is generally through the command line.

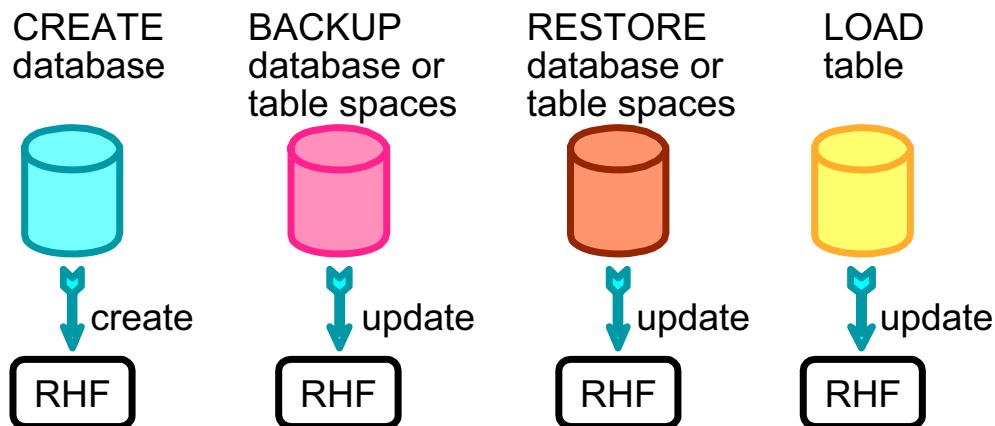
Since there are many configuration parameters, it is generally useful to filter the results of the GET CFG commands. This can be done with one of two operating system utilities depending on the platform where you are using the command:

- find — on Windows
- grep — on Linux and the various UNIX platforms

The search can be done case-insensitive (“/i” on Windows, “-i” on Linux/UNIX).

Of course, not all configuration parameters for logging will have *log* in the configuration list, and some irrelevant information may be listed.

Recovery history file



- The part of the database that was copied and how
- The time the copy was made
- The location of the copy
- The last time a restore was done

→ **db2 LIST HISTORY ALL FOR MUSICDB**

© Copyright IBM Corporation 2007

Figure 7-18. Recovery history file

CF238.3

Notes:

A recovery history file contains historical information for a database. The information provided in the chart above is not a complete listing of all the information provided in the history file. 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.

A recovery history file is created with each database and is automatically updated whenever:

- A database or table spaces are backed up
- A database or table spaces are restored
- A database or table spaces are rolled forward
- A table space is created
- A table space is altered
- A table space is quiesced
- A table space is renamed
- A table space is dropped
- A table is loaded

- A table is dropped
- A table is reorganized

You can use the summarized backup information in this file to recover all or part of a database to a given point in time. The information in the file includes:

- An identification (ID) field to uniquely identify each entry
- The part of the database that was copied and how
- The time the copy was made
- The location of the copy (stating both the device information and the logical way to access the copy)
- The last time a restore operation was done
- The time at which a table space was renamed, showing the previous and the current name of the table space
- The status of the backup operation: active, inactive, expired, or deleted
- The last log sequence number saved by the database backup or processed by a roll-forward recovery
- roll-forward recovery

The command shown in the visual will list all history for the *musicdb* database. If the word **backup** was used instead of **history**, only backups and restores would be shown.

An option on the RESTORE command permits only the recovery 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.

The default retention period for the history file is 366 days, but this may be overridden by using the database configuration parameter REC_HIS_RETENTN. The maximum value is 30000. You can also manually remove old entries from the file by using the PRUNE command. The list history command syntax is:

```
LIST HISTORY [ backup | rollforward | reorg | create tablespace  
            alter tablespace | dropped table | load | rename tablespace  
            archive log ]  
[ALL | SINCE timestamp |  
 CONTAINING [ schema.objectname | objectname ] ]  
FOR [DATABASE] database-alias
```

LIST BACKUP can be used as an alternative to LIST HISTORY BACKUP.

Examples of LIST HISTORY command:

```
db2 list history containing mel.table50 for musicdb  
db2 list backup containing userspace1 for musicdb  
db2 list history since 20040302 for musicdb  
db2 list backup since 20041012 for musicdb  
db2 list history all for musicdb  
db2 list backup all for musicdb
```

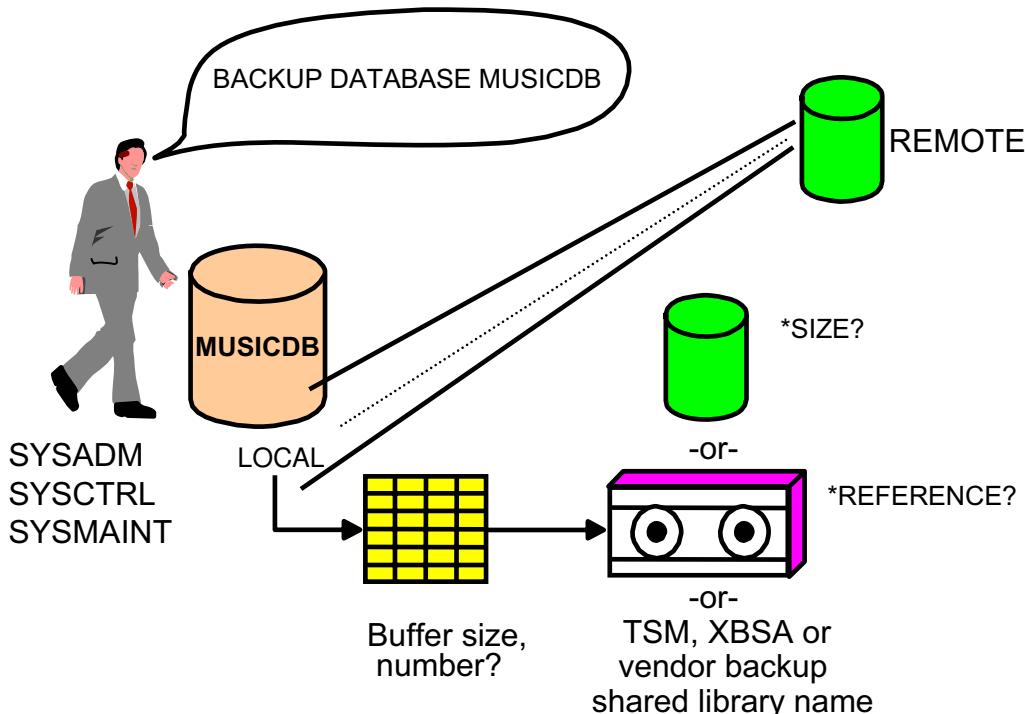
The PRUNE HISTORY command may be used to delete entries from the recovery history file. A minimum of yyyy may be specified as the timestamp. All entries with timestamps equal to or less than the timestamp provided are deleted from the recovery history file. The WITH FORCE OPTION specifies that the entries will be pruned according to the timestamp specified, even if some entries from the most recent restore set are deleted from the file.

Examples are:

```
db2 prune history 20040228100922  
db2 prune history 20040228 with force option
```


7.3 Backup and restore utilities

BACKUP command



© Copyright IBM Corporation 2007

Figure 7-19. BACKUP command

CF238.3

Notes:

Your planning considerations should include:

- You must have SYSADM, SYSCTRL, or SYSMAINT authority to use the BACKUP DATABASE command.
- The database may be local or remote. The backup image remains on the database server unless a storage management product such as Tivoli Storage Manager (TSM) is used.
- You can back up a database or table space to a directory, a tape, or a location managed by the TSM utility, to an XBSA interface, or to another backup vendor (if they provide the shared library).
- TSM allows you to specify that the backup is to use Tivoli Storage Manager (formerly ADSM) output. XBSA specifies that the XBSA interface is to be used.
- Backup Services APIs (XBSA) are an open application programming interface for applications or facilities needing data storage management for backup or archiving

purposes. Legato NetWorker is one of several storage manager that currently supports the XBSA interface.

- When backing up a database, you can indicate the amount of memory to be used for an internal buffer area on the backup command. If the command does not specify a BUFFER size, 1024 pages will be used.
- If you change the database configuration file to enable roll forward recovery (archive logging) by updating either logretain or userexit, you must take an offline backup of the database before it is usable.

If you are not able to make a backup copy of the database (because of time constraints) but need to use the database, you may change the database configuration file to disable roll forward recovery before the first backup. This means you are abandoning your attempt to implement log retention logging.

- A table space backup and a table space restore cannot be run at the same time, even if the backup and restore cover different table spaces.
- If you have tables that span more than one table space, you should back up (and restore) the set of table spaces together. This will eliminate some of the complexity of your recovery strategies.
- 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, 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 table space.

Syntax of the BACKUP command



Creates a backup copy of a database or table spaces

```
BACKUP DATABASE database-alias [USER username [USING password]]
[TABLESPACE (tblspace-name [ {,tblspace-name} ... ])] [ONLINE]
[INCREMENTAL [DELTAS]]
[USE {TSM | XBSA} [OPEN num-sess SESSIONS]
[OPTIONS {options-string | options-filename}]
| TO dir/dev
[ {,dir/dev} ... ] | LOAD lib-name [OPEN num-sess SESSIONS]

[OPTIONS {options-string | options-filename}]]
[WITH num-buff BUFFERS] [BUFFER buffer-size] [PARALLELISM n]
[COMPRESS [COMPRLIB lib-name [EXCLUDE]] [COMPROPTS options-string]]
[UTIL_IMPACT_PRIORITY [priority]] [{INCLUDE | EXCLUDE} LOGS] [WITHOUT
PROMPTING]
```

© Copyright IBM Corporation 2007

Figure 7-20. Syntax of the BACKUP command

CF238.3

Notes:

Consult the *Command Reference* for more detailed syntax of the BACKUP command.

The following considerations are useful when running the BACKUP DATABASE command:

- You must start the database manager (DB2START) before running the BACKUP command or API. When using the Control Center, you do not need to explicitly start the database manager.
- When using the command, API, or task under the Control Center, you must specify a database alias name, not the database name itself.
- A table space backup can include a single table space or multiple table spaces.
- You may specify multiple target areas (directory or tape devices) if you wish to take advantage of parallel backup. Consult the *Data Recovery and High Availability Guide and Reference* for details.
- The LOAD option is used to interface with vendor backup and restore I/O functions. Consult the *Data Recovery and High Availability Guide and Reference* for details.

- During the backup procedures, 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 can 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.

You may specify the number of pages to use for the backup buffers when you invoke the BACKUP DATABASE command. If no value is specified, a buffer size of 1024 pages will be used. If there is not enough memory available to allocate the buffer, an error will be returned.

The ideal backup buffer size should be a multiple of the extent size for the table spaces. If you have multiple table spaces with different extent sizes, specify a value that is a multiple of the largest extent size.

If any of the table spaces in a database is in an *abnormal* state, you cannot back up the database.

- If a system crash occurs during a critical stage of backing up a database, you cannot successfully connect to the database until you reissue the BACKUP DATABASE command.
- The BACKUP DATABASE command provides a concurrency control for multiple processes that are making backup copies of different databases. The control keeps a container open (on the backup target device) until the entire backup process has ended. If an error occurs during a backup process, the open container cannot be closed. With the container open, other backup processes to the same target drive may receive access errors. To correct any access errors, you must completely exit the backup process that caused the error and disconnect from the target device.
- If you are using the BACKUP DATABASE command for concurrent backup processes to tape, ensure that the processes do not target the same tape.
- The decision to select which table spaces to back up will also reduce the time during which the database is not available.
 - Long field and large object (LOB) data for a table must be placed in the same table space.
 - LONG FIELD and LOB data can be in a separate table space from the rest of the table data.
 - INDEX data could also be kept in a separate table space from the rest of the table data.

Backup strategies could be planned to take advantage of table space backup, thereby shortening the time for the backup to complete.

It is also possible to group several related tables together so that the table spaces 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.

Backup can be invoked locally or remotely. However, the backup image must be directed to local devices, unless TSM or another vendor product is used.

To reduce the amount of time required to complete a backup:

- Increase the backup buffer size
- Increase the number of buffers
- Use multiple target devices.

Reducing the amount of time naturally increases the CPU and I/O load on your system and will impact users on the system.



Note

Throttling back CPU & I/O intensive commands and utilities

If you do not want to impact users on the system, you may want to throttle the backup process. This will lengthen the time taken to complete the backup, but perhaps notably so. When throttling is in effect — typically with a constraint of 10% — online users will generally not notice the effect of backup.

We will discuss throttling commands and utilities such as this in a later unit.

The following parameters reference the visual:

database-alias

Specifies the alias of the database to back up.

USER *username*

Identifies the user name under which to back up the database.

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*

A list of names used to specify the table spaces to be backed up.

ONLINE

Specifies online backup. The default is offline backup. Online backups are only available for databases configured with logretain or userexit enabled. During an online backup, DB2 obtains IN (Intent None) locks on all tables existing in SMS table spaces as they are processed and S (Share) locks on LOB data in SMS table spaces.

USE TSM

Specifies that the backup is to use Tivoli Storage Manager output.

USE XBSA

Specifies that the XBSA interface is to be used. Backup Services APIs (XBSA) are an open application programming interface for applications or facilities needing data storage management for backup or archiving purposes.

OPEN *num-sessions* SESSIONS

The number of I/O sessions to be created between DB2 and TSM or another backup vendor product. This parameter has no effect when backing up to tape, disk, or other local device.

TO *target-area*

A list of directory or tape device names.

LOAD *library-name*

The name of the shared library (DLL on Windows operating systems) containing the vendor backup and restore I/O functions to be used. It can contain the full path. If the full path is not given, it will default to the path on which the user exit program resides.

WITH *num-buffers* BUFFERS

The number of buffers to be used. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value. However, when creating a backup to multiple locations, a larger number of buffers can be used to improve performance.

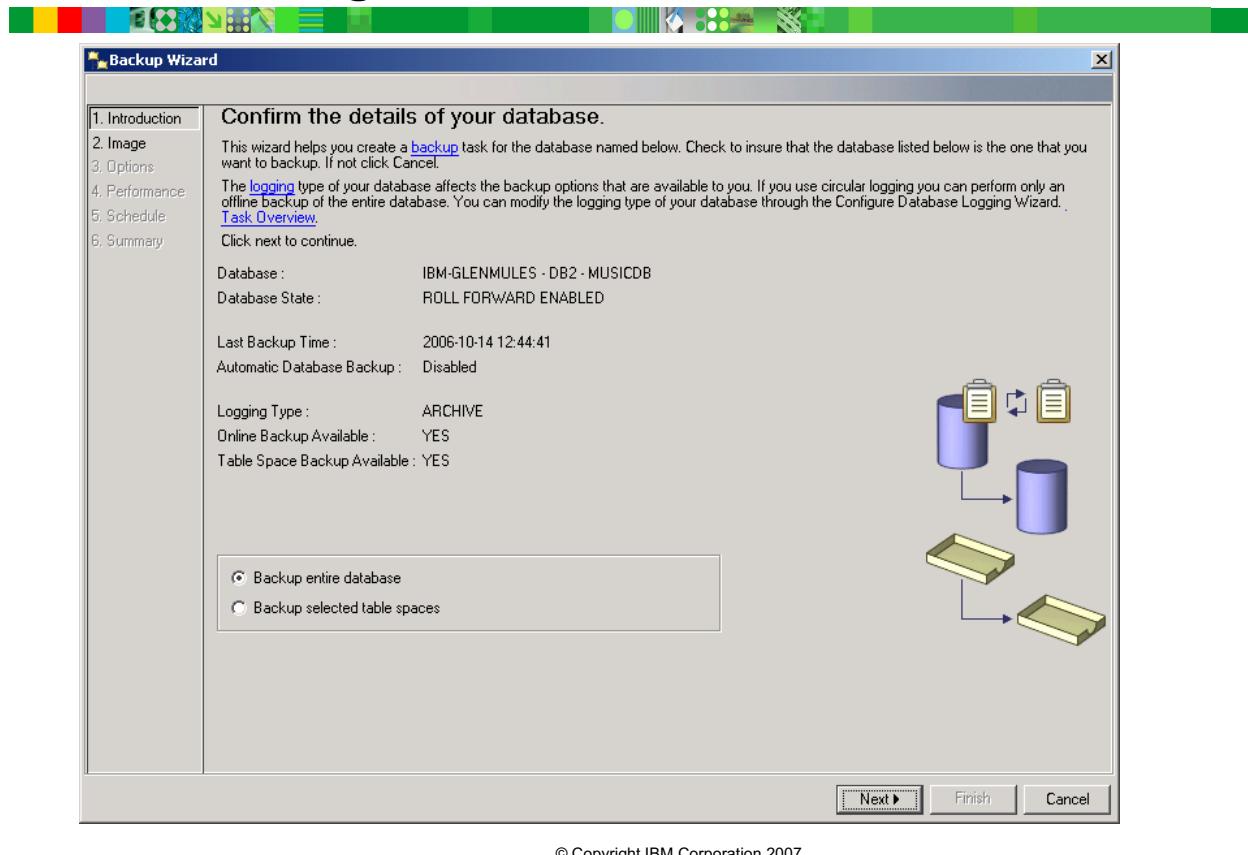
BUFFER *buffer-size*

The size, in 4 KB pages, of the buffer used when building the backup image. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value. The minimum value for this parameter is 8 pages.

WITHOUT PROMPTING

Specifies that the backup will run unattended, and that any actions which normally require user intervention will return an error message.

BACKUP using the GUI interface



© Copyright IBM Corporation 2007

Figure 7-21. BACKUP using the GUI interface

CF238.3

Notes:

To access the Backup Database GUI:

- Right-click and select the database
 - Left-click and select **Backup**

Use the Backup wizard to back up the objects in a database or database partition. First you specify if you want to back up the entire database or database partition, or if you want to back up only selected table spaces or table space partitions; then you make more advanced choices.

Introduction — This page lists the object that you have selected to back up. Verify that it is correct. If you are using archive logging, you can also choose whether you want to back up the entire database or back up only selected table spaces.

Note: Backing up all table spaces is not equivalent to backing up the entire database.

Table Spaces — This page is available only if you are in archive logging mode and you have chosen to back up selected table spaces. Use this page to specify the table spaces you want to back up. All non-temporary table spaces in your database are eligible to be

backed up, and are listed in the left-hand table. If you launched this wizard from one or more table space objects, they will be preselected in the right-hand table.

Image — Use this page to specify where you want to store your backup image. Indicate the media type you want to use and information about the particular media type.

Options — Use this page to specify whether you want to back up all data, back up data that has changed since the last full backup, or back up data that has changed since the last backup of any type. You can also specify whether you want the backup to occur while the database is online or offline. If the backup will be run offline, you can indicate that you would like to run a quiesce database command before the backup operation is run. You can also specify if you want (for an online backup) to include the relevant log files.

Note: If you are using circular logging, you can perform only an offline backup.

Performance — On this page, the wizard suggests recommended values for parallelism, the number of buffers, and the size of each buffer. You can modify these values. If you change the values on this page, you should click **Recommend** to have the wizard recalculate the recommended values.

Schedule — Use this page to indicate whether you want to run the backup operation immediately, or if you want to schedule the operation to run later. If you decide to run it later you have also the possibility to decide to run it once or repetitively, and to schedule it according your needs (automatic backup).

Summary — This page allows you to review the decisions that you made for your backup operation, and to view the command that will back up the objects in your database.

Online versus offline BACKUP

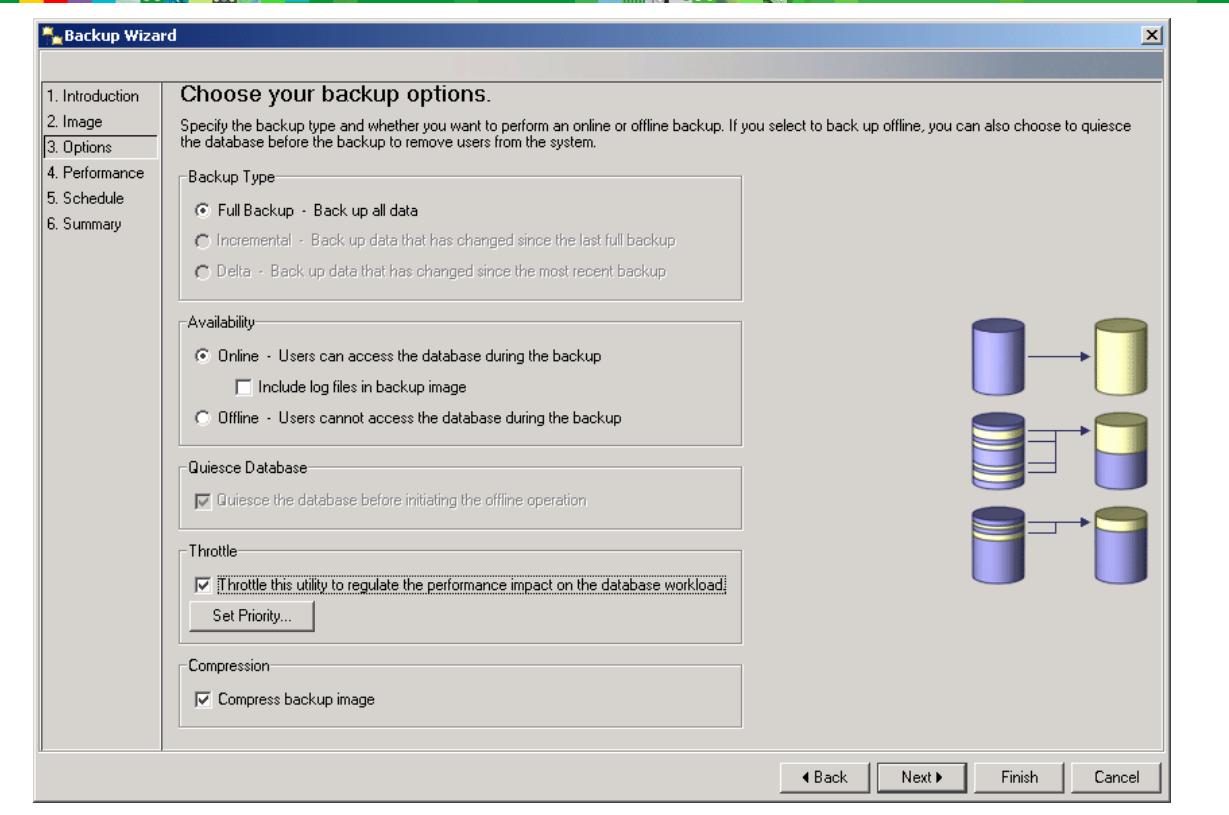


Figure 7-22. Online versus offline BACKUP

CF238.3

Notes:

You can do a backup while the database is either offline or online.

- If the backup is performed offline, only the backup task can be connected to the database. The stored data is consistent, and can be used for a restore only recovery or a restore followed by a roll forward.

The implication of offline backup is that the rest of your organization cannot connect to the database while the backup task is running.

- An offline backup is the only backup supported when circular logging is being used.
- 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 archive logging is enabled. The backup will not be valid for recovery purposes if you do not retain the log files written while the backup is taken. At the end of an online backup, the current active log is truncated (by default) and made available for offline archival.

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.

The backup files



- File name for backup images *on disk* has:
 - Database alias
 - Type of backup (0=full, 3=table space, 4=copy from table load)
 - Instance name
 - Node number
 - Catalog node number
 - Timestamp of backup (YYYYMMDDHHMMSS)
 - Sequence number (for multiple files)

MUSICDB.0.DB2.NODE0000.CATN0000.20061013150616.001
- Exact naming convention may differ slightly by platform
- Tape images are not named, but internally contain the same information in the backup header for verification purposes
- Backup history provides key information in easy-to-use format

© Copyright IBM Corporation 2007

Figure 7-23. The backup files

CF238.3

Notes:

Backup images consist of:

Database alias	A 1- to 8-character database alias name that was specified when the backup utility was invoked.
Type of backup	Type of backup operation, where: 0 represents a full database-level backup, 3 represents a table space-level backup, and 4 represents a backup image generated by the LOAD...COPY TO command.
Instance name	A 1- to 8-character name of the current instance that is taken from the DB2INSTANCE environment variable.
Node number	The database partition number. In single partition database environments, this is always NODE0000. In partitioned database environments, it is NODExxxx, where xxxx is the number assigned to the database partition in the db2nodes.cfg file.

Catalog node number	The database partition number of the catalog partition for the database. In single partition database environments, this is always CATN0000. In partitioned database environments, it is CATNxxxx, where xxxx is the number assigned to the database partition in the db2nodes.cfg file.
Timestamp of backup	A 14-character representation of the date and time at which the backup operation was performed. The time stamp is in the form yyyyymmddhhnnss, where: yyyy represents the year (1995 to 9999) mm represents the month (01 to 12) dd represents the day of the month (01 to 31) hh represents the hour (00 to 23) nn represents the minutes (00 to 59) ss represents the seconds (00 to 59)
Sequence number	A 3-digit number used as a file extension (for multiple files).

The recovery history file is updated automatically with summary information whenever you carry out a backup or restore of a full database or table space. The history file is usually 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 `db2ckbkp` utility allows you to display information regarding the backup images. It allows you to:

- Test the integrity of a backup image and determine whether it can be restored or not.
- Display the metadata about the backup that is stored in the backup header.

You must have read permissions on the backup images that you specify when using this utility. Refer to the *Data Recovery and High Availability Guide and Reference* for additional information on this utility.

Assume that a database backup of the database **MUSICDB** in the instance **INST00** was taken on **Friday, October 13, 2006 at 15:06:16**.

On Windows and Linux/UNIX, the naming convention would result in the following:

Alias	Instance	Cat	Node	Month	Hour	Second
musicdb.0.inst00.NODE0000.CATN0000.20061013150616.001						
Type	Node #	Year	Day	Minute	Sequence	

The type of backup taken is encoded:

- **0** is for full database
- **3** is for table space

- 4 is for copy from a table load

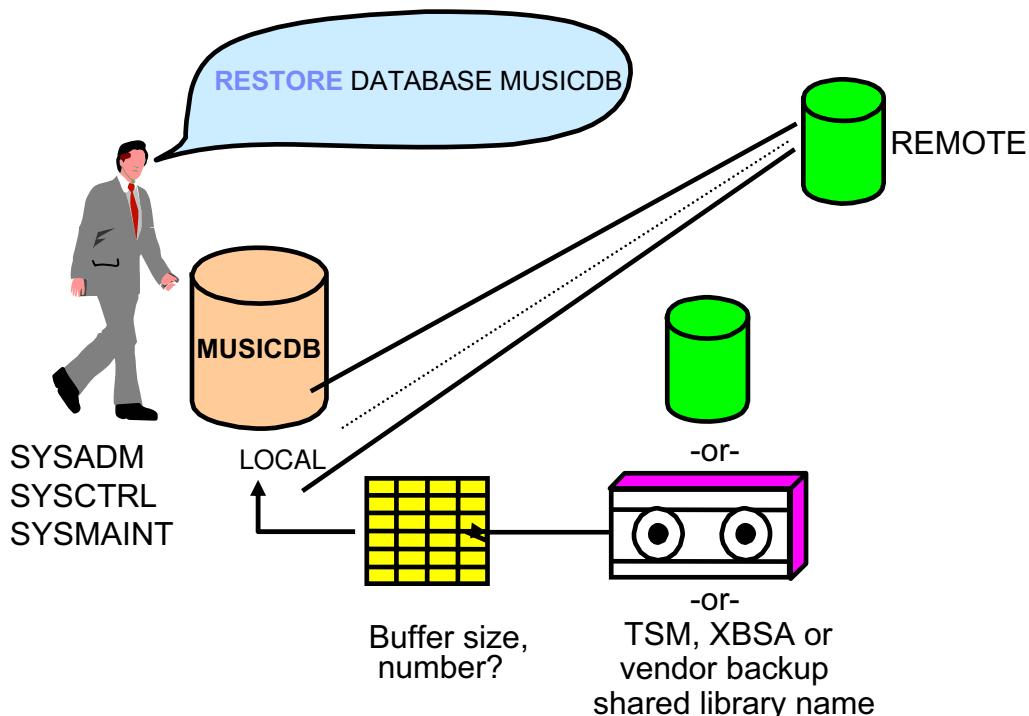
**Note**

Since operating systems do not share identical conventions for naming files, DB2 does not necessarily make the backup file names identical across platforms. However, the names are very easy to decode. Prior to DB2 9, and on Intel only, the naming convention would have resulted in the following:

Alias	Instance	Cat	Node	Month	Hour	Second
MUSICDB.0\INST00\NODE0000\CATN0000\20040701\131259.001						
Type	Node #	Year	Day	Minute	Sequence	

In this approach, a series of folders were used as part of the naming convention. With DB2 9, the naming conventions used for Intel and Linux/UNIX are *now* the same.

RESTORE command



© Copyright IBM Corporation 2007

Figure 7-24. RESTORE command

CF238.3

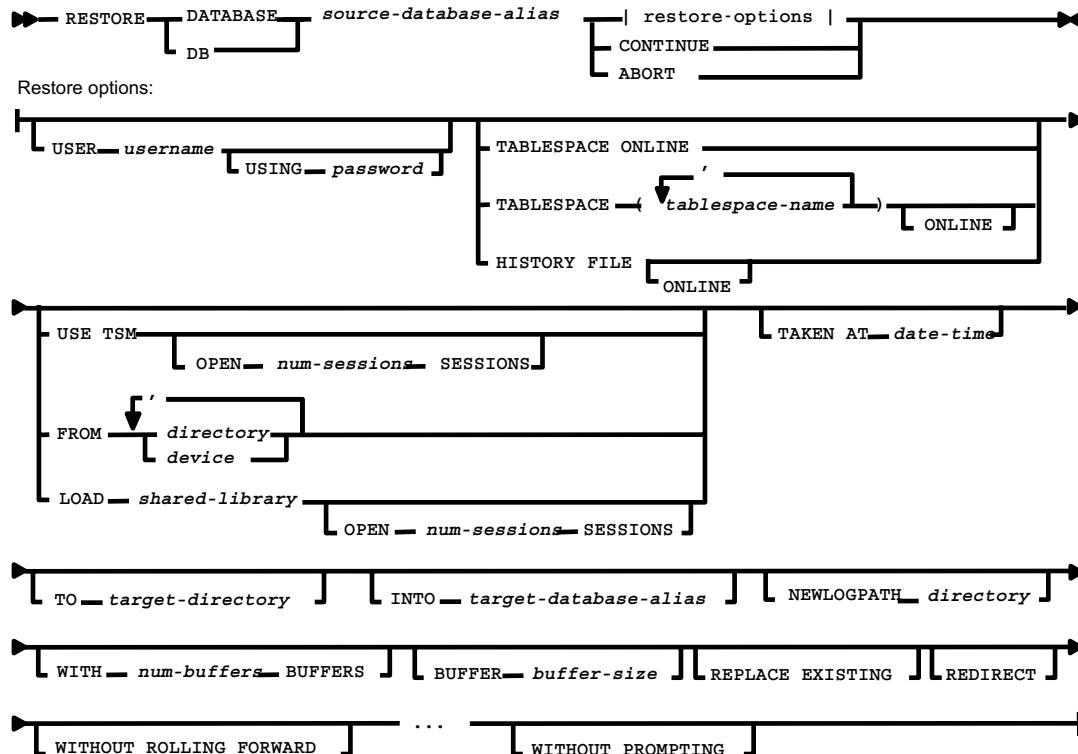
Notes:

When restoring a database, you should consider the following:

- You must have SYSADM, SYSCTRL, or SYSMAINT authority to restore to an existing database from a database or table space backup.
 - * SYSMAINT cannot restore to a new database. To accomplish this, you must have SYSADM or SYSCTRL authority.
- You can only use this command if the database or table space has been previously backed up with the BACKUP DATABASE command.
- You can choose to restore only the recovery history file, one or more table spaces, or everything in your backup image.
- A table space restore is particularly helpful if an error occurs in the table space containing the system catalog tables. An error in this table space will prevent users from connecting to the database, and having a backup containing only this table space could reduce your recovery time significantly.

- The RESTORE DATABASE command can use the Tivoli Storage Manager (TSM) utility, and any restrictions of that utility should also be considered.
- Another vendor product, or the XBSA interface, may also be used if that option 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.
- A table space restore can be online (share mode) or offline (exclusive mode). An online restore permits access to **OTHER** table spaces, but still requires exclusive use of the target table spaces of the restore.
- The size and number of the buffers 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. If the command does not specify a BUFFER size, a buffer size of 1024 pages will be used.
- The database may be local or remote.

Syntax of the RESTORE command



© Copyright IBM Corporation 2007

Figure 7-25. Syntax of the RESTORE command

CF238.3

Notes:

Consult the *Command Reference* for more detailed syntax of the RESTORE command.

The following considerations are useful when running the RESTORE command:

- The database manager must be started before restoring a database.
- The database to which you restore the data may be the same as the one from which the data was originally backed up, or it may be different. You may restore the data to a new or an existing database.

If you are restoring the database to a new database, but the containers for the table spaces do not yet exist, you will want to do a redirected restore (RESTORE ... REDIRECT). This will restore the definitions for the table space containers, and allow you to modify the definitions before continuing the restore (RESTORE ... CONTINUE). You may also want to use NEWLOGPATH to indicate where the log file directory should be placed.

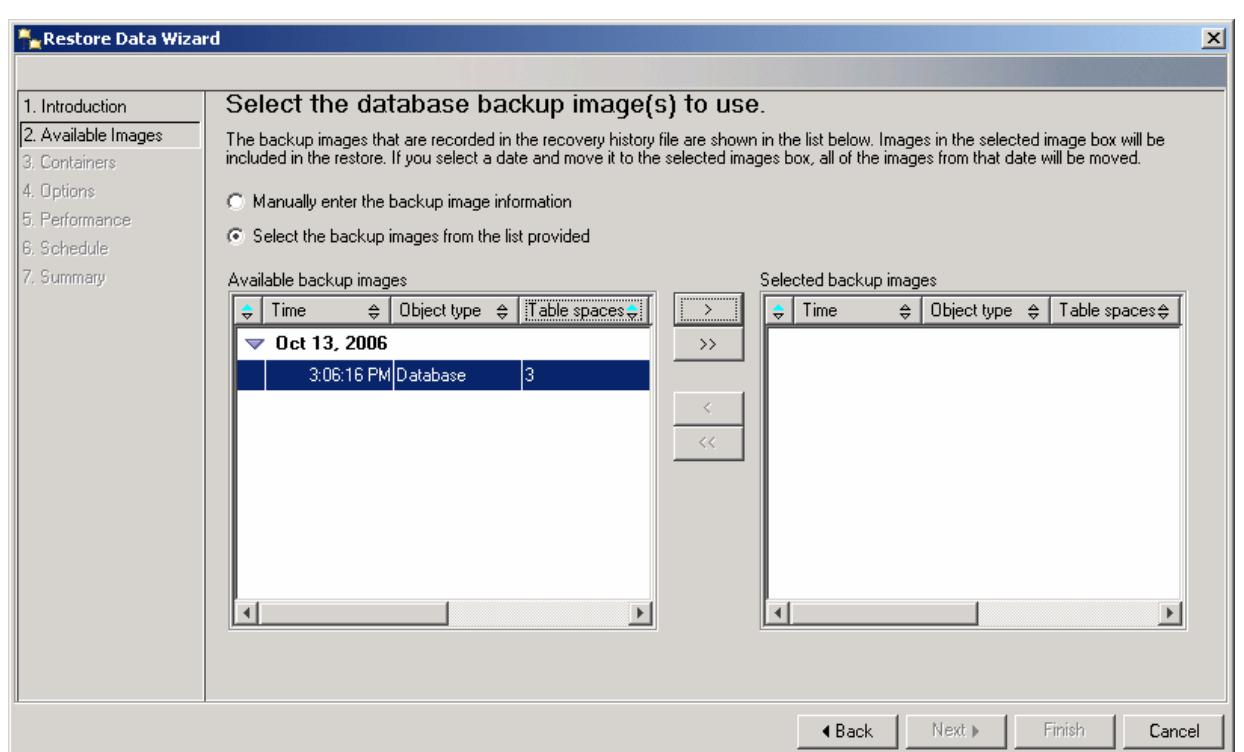
If the database into which you are restoring already exists, you can avoid the warning message that requires a response before overwriting the database by indicating REPLACE EXISTING.

- During the restore process, you can optionally select to use multiple buffers to improve the performance of the restore utility. The multiple internal buffers may be filled with data from the backup media. The number of buffers you allocate should be at least twice the number of the 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. If you do not specify the number of pages, each buffer will have a size of 1024 pages. If there is not enough memory available to allocate the buffer, an error will be returned.

- The backup copy of the database or table space to be used by the RESTORE DATABASE command can be located in a directory, a tape, or a location managed by the Tivoli Storage Manager (TSM) utility, to an XBSA interface, or to another vendor (if they provide the shared library).
- The LOAD option is used to interface with vendor backup and restore I/O functions. Consult the *Data Recovery and High Availability Guide and Reference* for details.
- Once the RESTORE DATABASE command starts for a database restore, 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.
- Once the RESTORE DATABASE command starts for a table space restore, the table space is not usable until the RESTORE command followed by a roll forward recovery completes successfully.
- If a system failure occurs during any stage of restoring a database, you cannot connect to the database until you reuse the RESTORE DATABASE command and successfully complete the restore.
- Individual table spaces can be restored from either a database or a table space backup.
- A table space restore using the ONLINE parameter allows other table spaces to be used while the restore is running.
- If a system failure occurs during the restoring of a table space backup, only the table space being restored is unusable. The other table spaces in the database can still be used.
- TABLESPACE (tablespace-name) allows specification of a subset of the table spaces in the backup image to be restored.
- If the history file for the database has been deleted or corrupted, you can restore just the history file from the backup image (HISTORY FILE).

RESTORE using the GUI interface



© Copyright IBM Corporation 2007

Figure 7-26. RESTORE using the GUI interface

CF238.3

Notes:

To access the Restore Database GUI:

- Right-click and select the database
 - Left-click and select **Restore**

Use the Restore wizard to perform any of the following tasks:

- Restore a database or database partition
- Roll forward a database or database partition
- Restore a database backup image to a new database
- Restore the history file for a database
- Restore a table space or table space partition
- Roll forward a table space or table space partition

First, you specify the objects you want to restore, and select the image that you want to use; then you make more advanced choices.

Introduction — This page lists the objects that you have selected to restore. Verify that they are correct. Also use this page to specify whether you are restoring to an existing database, restoring to a new database, or restoring the history file for the database.

Note that, if you have log archiving enabled, the summary will indicate that you can restore to the last transaction. If you are using circular archiving, certain restore options will not be available.

Restore Objects — This page is available only if you are restoring to an existing database and if you are in log archiving mode. If you only want to restore a subset of the database, use this page to specify the table spaces you want to restore.

Restore to a New Database — This page is available only if you have chosen on the Introduction page of this wizard to restore to a new database. Use this page to specify the name and location of the new database.

Available Images — Use this page to select the database image or images that you want to use to restore your database.

Containers — Use this page to modify the table space containers used for your database. You can add new containers, change existing containers, or remove containers.

Roll Forward — This page is available only if you are in log archiving mode. On this page, you can set roll forward parameters and specify where your log files are already stored so that they can be retrieved and used in the roll forward operation.

Final State of Database — Use this page to specify the state the database should be put in when the restore has completed. You can leave the database in roll forward pending state so that you can continue to make changes to the restore, or you can return the database to active state so that it is usable.

Options — This page allows you to set various restore options. If you are restoring an archival database, you can set quiesce options.

History Options — This page is available only if you have selected to restore only the history file for your database. Use this page to select how you want to have the restore operation run: online or offline.

Performance — On this page, the wizard suggests recommended values for parallelism, the number of buffers, and the size of buffers. You can modify these values.

Schedule — Use this page to schedule when the restore operation is to run.

Summary — This page allows you to review the decisions that you made for your restore operation, and to view the command that will restore the objects in your database.

Backup/restore table space considerations

- Roll-forward must be enabled
- Can choose to restore a subset of table spaces
- Generally best to put multiple spaces in one backup image
 - Makes table space recovery strategy easier
 - Provides access to related tables spaces and coherent management of these table spaces
- Handling of long/LOB/XML data requires a correlated strategy
- Point-in-time recovery is supported, but has requirements
- Faster recovery for Catalogs using Tablespace Level backup
- Critical business application tables should obviously be the focus of the backup/restore, but other tables are needed in support of these tables

© Copyright IBM Corporation 2007

Figure 7-27. Backup/restore table space considerations

CF238.3

Notes:

In order to use table space level backup and restore, archive logging must be used.

A subset of table spaces can be restored from a database or table space backup image.

One of the key reasons to use table space level 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 table space, this would be likely to lead to a management problem concerning backup and recovery.

If the table spaces associated with closely related tables are all contained in a single backup, only the applications targeting the closely related tables are affected. In many cases, such applications would be affected even if a single table in the group was unavailable. This is especially true in the case of referentially constrained tables. You should consider grouping the table spaces that support referential structures in a single backup image.

The key is to reduce the amount of data involved in the event recovery; this is necessary while controlling the impact on management of the backup/restore strategy.

LOB data and long field data can be placed in a table space that is separate from the regular data. This is often desirable from a recovery standpoint because the frequency of taking backups of the LOB data can be much less than that of the regular data. The nature of LOB data is that it tends not to be updated frequently and it is large. However, if a REORG of such a table and its LOB data is one of the actions recorded in the log files through which you are rolling forward, you must have all table spaces relating to the table in the backup, defeating one of your reasons to separate the data. The solution is to establish new backups of the table spaces associated with such tables AFTER a REORG that includes the LOB data.

Point-in-time recovery during roll forward is supported. The LIST TABLESPACES command can be used to get a minimum roll forward time.

If the catalog tables are involved in a recovery situation, access to the entire database is impacted. Therefore, it is good practice to maintain table space level backups of your catalog tables. If you need to recover the catalog, the duration of the process will be reduced if you can restore a table space backup instead of a database backup.

Application tables considered critical to the business should also be considered prime candidates for table space recovery. The major reason is the reduction in downtime provided through the table space backup/restore strategy.

Roll forward pending



- *Roll forward pending* is set as a result of:
 - Restore of *offline* database backup omitting the command option **WITHOUT ROLLING FORWARD**
 - Restore of an *online* database backup
 - Restore of *any table space* level backup
 - DB2 detects media failure isolated at a table space
- Scope of pending state managed by DB2
 - *Database* in pending state will not permit any activity
 - *Table spaces* in pending state will permit access to other table spaces

© Copyright IBM Corporation 2007

Figure 7-28. *Roll forward pending*

CF238.3

Notes:

Roll forward pending is a state used by DB2 to protect the integrity of the database. This state indicates that a roll-forward process is necessary to ensure consistency of the data.

Roll forward pending can occur either at the database or at the table space level.

If the database is in a roll-forward pending state, no activity to the database is allowed. If a table space is in a roll forward pending state, only that table space is unusable.

A database will be put into a roll-forward pending state when:

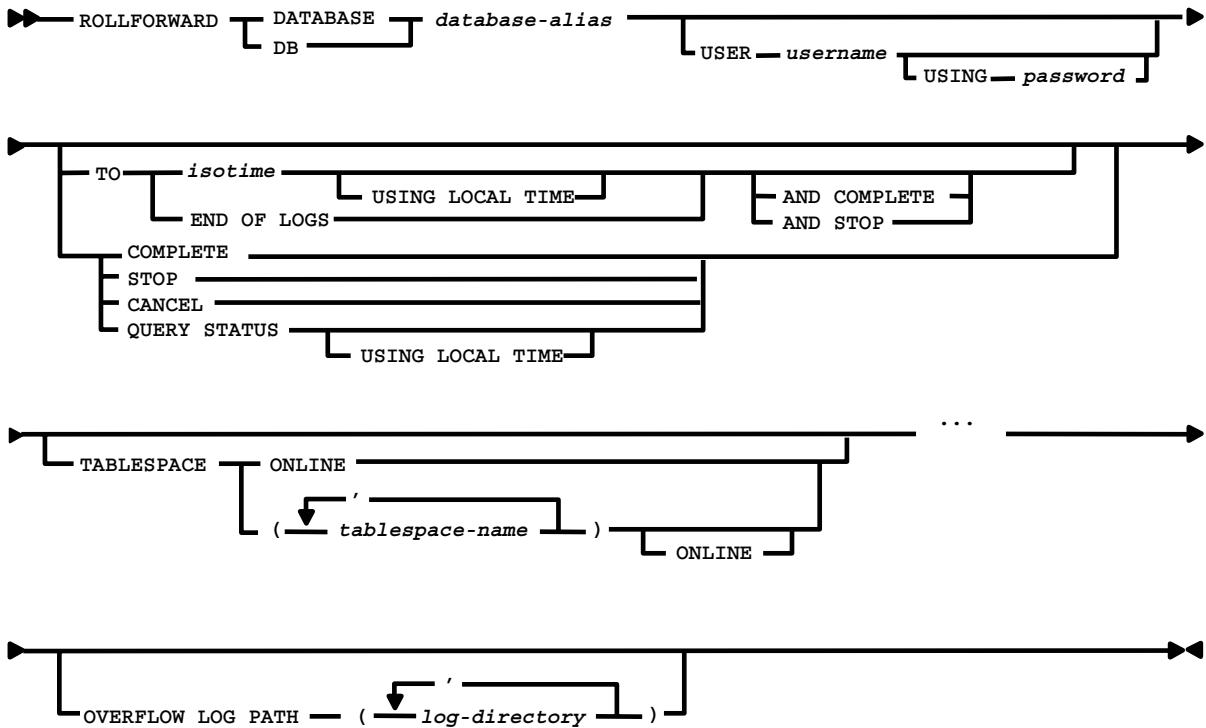
- Restoring an OFFLINE DATABASE backup and omitting the option WITHOUT ROLLING FORWARD. This applies only to a database using archive logging.
- Restoring an ONLINE DATABASE backup.

A table space will be put into a roll forward pending state when:

- A table space is restored.

DB2 detects a media failure and isolates it at the table space level.

Syntax of the ROLLFORWARD command



© Copyright IBM Corporation 2007

Figure 7-29. Syntax of the ROLLFORWARD command

CF238.3

Notes:

Authorization for this command requires SYSADM, SYSCTRL, or SYSMAINT.

ROLLFORWARD applies transactions recorded in the database log files.

The command needs to be invoked after a database or a table space backup has been restored, or if any table spaces have been taken offline by the database due to a media error.

Restore is the first phase of a complete roll forward recovery of a database or table space.

After a successful database restore, a database that was configured for roll forward recovery at the time the backup was taken enters a roll forward pending state, and is not usable until the ROLLFORWARD command has been run successfully. If the restore was for a table space, the table spaces restored are placed in a roll forward pending state.

When the ROLLFORWARD DATABASE command is issued, if the database is in a roll-forward pending state, the database is rolled forward. If the database is not in a roll forward pending state, all table spaces in the database in the roll forward pending state are processed.

Another database RESTORE is not allowed when the roll forward process is running.



Note

If you restored from a full **OFFLINE** database backup image, you can bypass the roll-forward pending state during the recovery process. The RESTORE DATABASE command gives you the option to use the restored database immediately **WITHOUT ROLLING FORWARD** the database.

You **CANNOT** bypass the roll-forward phase when recovering at the table space level or if you restore from a backup image that was created using the **ONLINE option** of the BACKUP DATABASE command.

Command Parameters

- DATABASE database-alias

The alias of the database to roll forward.

- USER username

Identifies the user name under which the database is to be rolled forward.

- USING password

The password used to authenticate the user name. If the password is omitted, the user is prompted to enter it.

- TO isotime

The point in time to which all committed transactions are to be rolled forward (including any transactions committed precisely at that time, as well as all transactions committed previously).

This value is specified as a timestamp, a seven-part character string that identifies a combined date and time. The format is *yyyy-mm-dd-hh.mm.ss.nnnnnn* (year, month, day, hour, minutes, seconds, microseconds), which is specified in Coordinated Universal Time (CUT). An additional option, USING LOCAL TIME, can be specified to indicate the time in local time of the server.

- TO END OF LOGS

Specifies that all committed transactions from all available active and archived log files should be applied to the database or tablespace that was restored.

- AND STOP / AND COMPLETE

Completes the roll forward recovery process by rolling back any incomplete transactions and turning off the roll forward pending state of the database. This allows access to the database once again. Both STOP and COMPLETE perform the same function.

- STOP / COMPLETE

Does not roll forward any more log records, and completes the roll forward recovery process by rolling back any incomplete transactions and allowing access to the database once again. Both STOP and COMPLETE perform the same function.

- CANCEL

Cancels the roll forward process and leaves the database or table spaces on which forward recovery has been started in restore-pending state.

- QUERY STATUS

Lists the log files that the database manager has rolled forward, the next archive file required, and the time stamp (in CUT) of the last committed transaction since roll forward processing began. The information returned contains the following fields:

- Roll forward status — Status may be database or table space roll forward pending, roll forward in progress, roll forward processing STOP, or no roll forward pending.
- Next log file to be read — A string containing the name of the next required log file.
- Log files processed — A string containing the names of the processed log files that are no longer needed for recovery, and that can be removed from the directory.
- Last committed transaction — A string containing a time stamp in ISO format (yyyy-mm-dd hh.mm.ss). This time stamp marks the last transaction committed after the completion of roll forward recovery.

Note: QUERY STATUS is the default if the TO, STOP, COMPLETE, and CANCEL clauses are omitted.

- TABLESPACE ONLINE

This keyword is specified to allow the table space level roll forward recovery to be done online. This means that other agents are allowed to connect and use other table spaces while roll forward recovery is in progress.

- TABLESPACE ONLINE tablespace-name

The table space name must be indicated for table space level roll forward to a point-in-time. This also allows a subset of table spaces to be specified for a roll forward to the end of logs.

- OVERFLOW LOG PATH log-directory

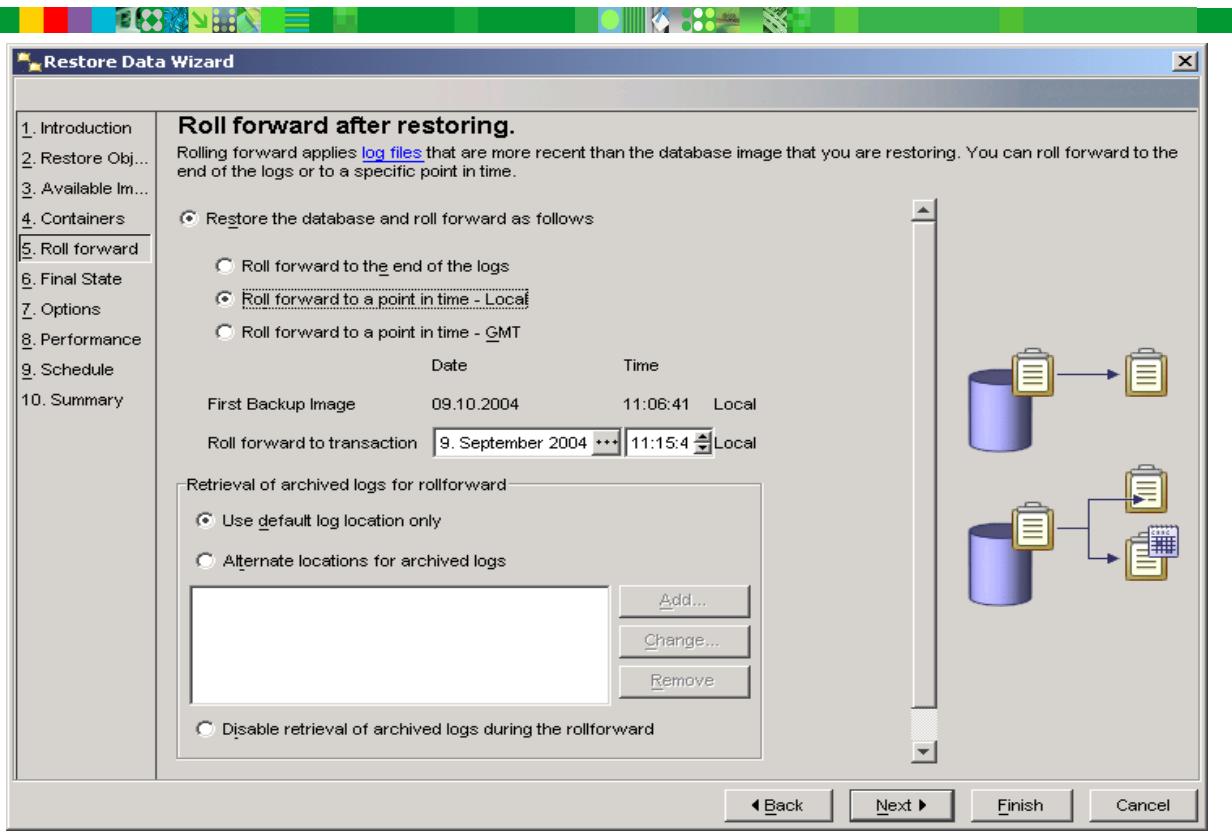
Specifies an alternate log path to be searched for archived logs during recovery.

If enabling an existing database for archive logging, 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 that 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 using archive logging.

Rolling databases and table spaces forward may involve prerequisites and restrictions that are beyond the scope of this course.

For more detailed information, refer to the *Data Recovery and High Availability Guide and Reference*.

ROLLFORWARD — GUI interface



© Copyright IBM Corporation 2007

Figure 7-30. ROLLFORWARD — GUI interface

CF238.3

Notes:

To access the Roll Forward Database GUI:

- Right-click and select the database
 - Left-click and select **Roll forward**

The roll forward GUI can also be accessed from the Restore Database Wizard. (The screen capture above is from the Restore Wizard.) The administrator can select roll forward to the end of logs or to a point in time. The point in time can be specified as local time on the client or on the server when using the GUI.

Use the Roll forward wizard to perform any of the following tasks:

- Roll forward a database or database partition
- Roll forward a table space or table space partition

Roll Forward — You will only be able to enter the Roll forward wizard if the object is in roll forward pending state. On this page, you can set roll forward parameters and specify where your log files are already stored so that they can be retrieved and used in the roll forward operation.

Final State of Database — Use this page to specify the state the database should be put in when the roll forward has completed. You can leave the database in roll forward pending state so that you can continue to apply logs to the restored image, or you can return the database to active state so that it is usable.

Summary — This page allows you to review the decisions that you made for your roll forward operation, and to view the command that will roll forward the objects in your database.

ROLLFORWARD — how far?



- End-of-logs
 - *End* means the end of the current log path
 - Other logs may need to be moved into the path
- Point-in-time (PIT)
 - Specified in Universal Time Coordinated (UTC) via command
 - Specified in local time *on server* with USING LOCAL TIME
 - Specified in local time on the client via GUI interface
 - Format: yyyy-mm-dd-hh.mm.ss.nnnnnn
- ONLINE backup requirements
 - Requires ROLLFORWARD past end of backup
 - Recovery history file (RHF) is useful
- Table space point-in-time considerations
 - Minimum roll forward time maintained for each table space — and the earliest possible point-in-time is the latest of the times in each of the relevant table spaces
 - Backup required after roll forward to establish a baseline for future backup/recovery

© Copyright IBM Corporation 2007

Figure 7-31. ROLLFORWARD — how far?

CF238.3

Notes:

The database administrator can clear a ROLLFORWARD PENDING condition by issuing the ROLLFORWARD command. The point in time to which the roll forward stage proceeds is also controllable by the administrator. An administrator can roll forward to end of logs or to a specific point in time. Use the Coordinated Universal Time (CUT) or local time on the server when roll forward is specified in a command. Use the CUT, local time on the server, or local time on the client when roll forward is specified through the GUI.

The integrity of the database must be protected, therefore the earliest point in time at which the roll forward stage can end is the end of the online backup image.

Table space point-in-time recovery must protect the integrity of the relationships that exist between tables in the table spaces. These relationships include referentially constrained tables and single tables that have objects contained in multiple table spaces. A minimum roll forward time is kept at the table space level and can be displayed with the LIST TABLESPACES command. A backup is required following a point-in-time recovery of a table space.

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 coordinated universal time (CUT) or local time of the server.



Important

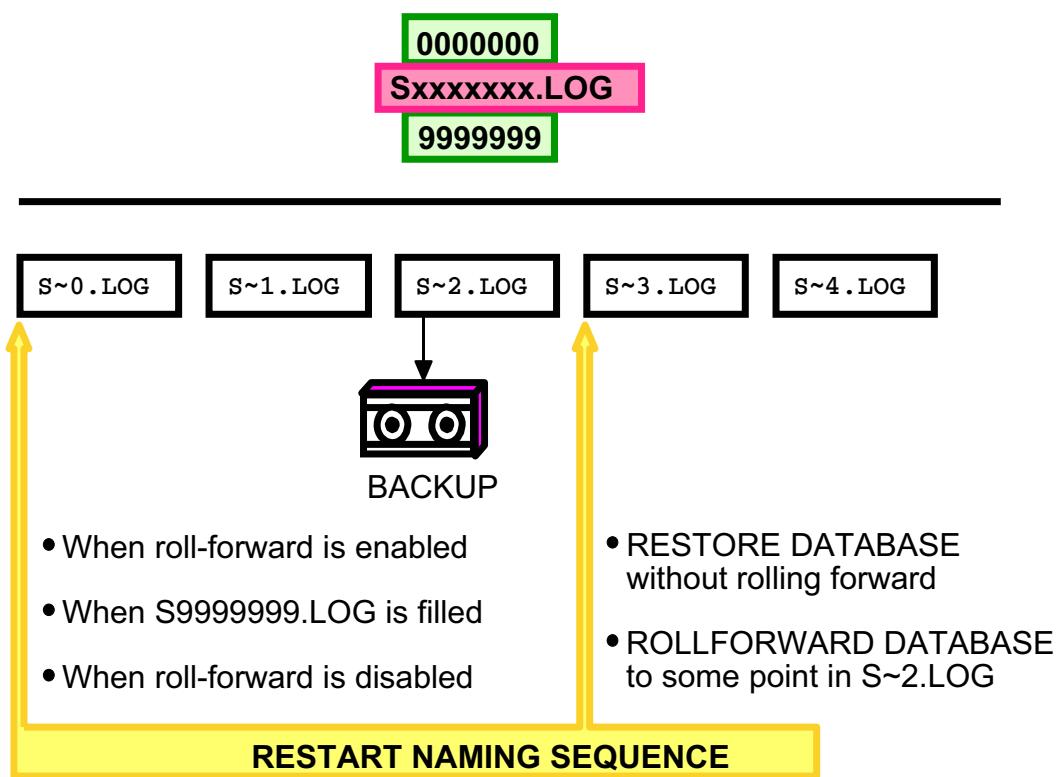
The coordinated universal time is used in log records so that the database manager does not have a recovery dependency regarding daylight savings time or other local time anomalies. However, this introduces a degree of complexity for the database administrator. While backup images are identified via timestamps reflecting local time, rolling forward (since it applies to logs) designates times in CUT format. If specifying USING LOCAL TIME, care must be taken to indicate the time correctly, particularly if daylight savings time changes are close to the time of the roll forward.

In many cases, recovery will be desired to the most current point possible. Therefore, the END OF LOGS option will be commonly used.

The administrator can STOP / COMPLETE the roll forward process and allow access to the database.

AND STOP / AND COMPLETE is a necessary parameter to permit the database manager to roll back any transactions that are not complete 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.

Log file naming



© Copyright IBM Corporation 2007

Figure 7-32. Log file naming

CF238.3

Notes:

The DB2 log file naming convention starts at **S0000000.LOG** and sequentially names logs until **S9999999.LOG** is reached. The higher numbers will only be used if archive logging is configured for the database. Otherwise, only the numbers corresponding to the actual number of primary and secondary logs for circular logging will be used.

The database manager will restart the naming sequence at a value different from **S0000000.LOG** after a successful RESTORE of a database configured for roll forward recovery. The actual name of the log file at which reuse starts depends on the recovery point. The log file that corresponds to the point of recovery is not reused. The next name will be reused. In the diagram above, a RESTORE WITHOUT ROLLING FORWARD to an OFFLINE backup created at the time the **S0000002.LOG** was active will cause **S0000003.LOG** to be the name at which reuse will start. The same name would also be reused if a ROLLFORWARD to a point in time within **S0000002.LOG** was completed.

Normally, the reuse of log names should not be a concern to the database administrator, since a restore with or without ROLLFORWARD establishes a new point of recovery. Therefore, log files associated with names greater than the point of recovery are not useful,

and reusing them is not a concern. An exception would be that the database administrator wants to retain the recoverability of multiple backups or the recoverability of a single backup to a later point in time. For example, assume that the backup in the above illustration is BACKUP1. Assume that a second backup, BACKUP2, was taken at some point that corresponds to the **S0000003.LOG**. If the database administrator restores to BACKUP1 and rolls forward to some point within **S0000002.LOG**, the name **S0000003.LOG** will be used as the next log after recovery. If the prior log of this name is not moved to a different directory, it will be overwritten during normal use of the database. Such reuse will make a future recovery to BACKUP2 (or BACKUP1 with a ROLLFORWARD to a point within the old **S0000003.LOG**) impossible. Again, this is not usually a concern because typical recovery strategies establish a new *starting* point after a restore is successful.

Disaster recovery considerations

Should you use database recovery, table space recovery, or a combination?

- Database recovery

- Offline backup does not require a roll forward
- Entire database can be restored on another system
- Reduces backup management

- Table space recovery

- Requires a roll forward
- Backup afterwards may be required
- Increases the number of backup images

© Copyright IBM Corporation 2007

Figure 7-33. Disaster recovery considerations

CF238.3

Notes:

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

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

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

The usefulness of a table space backup for disaster recovery depends on the scope of the failure. Typically, disaster recovery requires that you restore the entire database; therefore,

a full database backup should be kept at a standby site. Even if you have a separate backup image of every table space, you cannot use them to recover the database. If the disaster is a damaged disk, a table space backup of each table space on that disk can be used to recover. If you have lost access to a container because of a disk failure (or for any other reason), you can restore the container to a different location.

Both table space backups and full database backups can have a role to play in any disaster recovery plan. The DB2 facilities available for backing up, restoring, and rolling data forward provide a foundation for a disaster recovery plan. You should ensure that you have tested recovery procedures in place to protect your business.

Questions to consider

- Will the log files and the database files be on separate physical disks?
- What medium can best handle the volume of log data to be generated?
- Should copies of log files, backups, or both be made and managed?
- Can the OS support mirroring? Should a mirror log path be defined?
- How far back should recovery be enabled?
- When will all log files and backups be discarded?
- Can a user-exit sample be tailored or used?
- Are paging space/swap files on the same device or drive as DB2 logs?

© Copyright IBM Corporation 2007

Figure 7-34. Questions to consider

CF238.3

Notes:

No two installations will have identical recovery requirements, but the questions listed above are included to highlight some key areas that should be considered.

Particular consideration should be paid to the use of user-exit programs. Although these are supplied by IBM as samples, and are therefore not guaranteed regarding functionality, they can be good starting points. It is necessary to test the functionality at your installation. Consult the *Data Recovery and High Availability Guide and Reference* for additional details.

Logging/backup requirements summary



Consideration	CRASH	VERSION	ROLL-FORWARD
Logging	Circular or Archive	Circular or Archive	Archive
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
Table Space Recovery	N/A	NO	YES

© Copyright IBM Corporation 2007

Figure 7-35. Logging/backup requirements summary

CF238.3

Notes:

The table above summarizes the key points regarding backup and logging. The descriptions of resources and management are relative. For example, some database administrators may consider management of an installation's roll forward recovery strategy quite simple, once the planning and implementation process has been completed. However, compared to supporting crash recovery, the cost is high.

Advanced recovery features

- On-demand log archiving
- Block transactions on log directory full
- Split mirror backup and suspended I/O
- Cloning a database
- Split mirror as standby database
- Split mirror as a backup image
- Incremental and delta backup
- Backup to named pipe
- Parallel rollforward recovery
- Relocating a database or a table space

© Copyright IBM Corporation 2007

Figure 7-36. Advanced recovery features

CF238.3

Notes:

DB2 advanced recovery and logging features include:

- **On-demand log archiving**

You can close and archive the active log at any time. For recoverable databases, this feature lets you create a complete set of log files to a specific time for such purposes as offsite storage. To do so, use the following command:

```
ARCHIVE LOG FOR DATABASE sample USER  
name USING password
```

If other applications have transactions in progress, you'll notice a slight performance degradation as the ARCHIVE LOG command flushes the log buffer to disk. If user exits are enabled, an archive request is issued after the logs are closed and truncated. Completion of the archive command does not guarantee that the logs have been moved to the archive directory, so you should check to make sure.

- **Block transactions on log directory full**

In earlier versions, DB2 would crash if the log directory was full and a new log file could not be created. In this version, DB2 will attempt to create the log every five minutes, while reporting the condition to the db2diag.log. This feature is controlled by the database configuration option BLK_LOG_DSK_FUL.

This feature allows read-only applications to continue, provided that they are not dependent on rows locked by an update transaction. Applications that are updating data will not complete their operation until the log has been created.

- **Split mirror backup and suspended I/O**

To meet the need for increased availability, the following backup and recovery options may be implemented. Backup on very large databases requires substantial time and resources, and can interfere with continuous availability. Many customers cannot afford offline or online backups on a 1 TB live database.

Newer disk drive technology provides opportunities for high availability through fault tolerance and redundancy.

The online split mirror function allows a consistent mirror of a database to split off while update processing is temporarily suspended using the SET WRITE SUSPEND FOR DATABASE command. Recovery on the split copy is acceptable for consistency; backups and system copies can be done from a mirror image.

The db2inidb utility must be used on the split copy to indicate the intended use of the database. Use only logs from the primary database, not from the split image. You must run db2inidb on all partitions on a partitioned database environment before the split image can be accessed.

The command follows:

```
db2inidb <db_name> as snapshot  
db2inidb <db_name> as standby  
db2inidb <db_name> as mirror
```

Snapshot results in crash recovery, making the database consistent (a new log chain is started). You cannot roll forward through any of the logs from the original database. The database will be fully available at this time for any operation, including backup.

Standby places the database in a roll forward pending state; crash recovery is not performed, and the database remains in an inconsistent state with in-flight transactions still outstanding. This feature lets authorized users perform an offline backup of either the entire database or a subset of the table spaces.

- **Cloning a database**

A primary database can be cloned so that it can be used for read-only purposes such as running reports or for analysis purposes. The steps are as follows:

1. Suspend I/O on primary system: db2 set write suspend for database.

2. Split the mirror from the primary database using an operating system-level command.
3. Resume I/O on primary system: `db2 set write resume for database`. The database on the primary system should be back to a normal state.
4. Attach to the mirrored database from another machine.
5. Start database instance: `db2start`.
6. Start the DB2 crash recovery: `db2inidb <db_name> as snapshot`.

You can also clone a database for an offline backup, but you cannot use this backup to roll forward because the log chain will not match if restored on the primary system.

- **Split mirror as standby database**

For a standby database, the mirrored database is continually rolling forward through the logs. Even new logs created by the primary database are constantly fetched from the primary system. This results in higher availability and less recovery time. To execute, use the first four steps from the cloning process and add these steps:

5. Copy the logs: Set up a user exit program to retrieve log files from the primary system so that the latest logs will be available for this mirrored database.
6. Place the mirror in roll forward pending state, and roll forward the mirror: `db2inidb <db_name> as standby`. Remove the suspend write state and roll forward the database to the end of the logs.
7. Continue this process (copy and roll forward) until the primary database is down.

- **Split mirror as a backup image**

To use the mirrored system as a backup image to restore over the primary system:

1. Copy over the image: Use operating system commands to copy the mirrored data and logs on top of the primary system.
2. Start the database instance (`db2start`).
3. Place the mirror in roll forward pending state, and roll forward the mirror: `db2inidb <db_name> as mirror`.
4. Remove the suspend write state and roll forward the database to end of logs. The logs from the primary database must be used when rolling forward: `db2 rollforward database <db_name> to end of logs`.

- **Incremental and delta backup**

DB2 supports two types of smaller backups: incremental and delta. These backups provide the benefits of a smaller backup size. With a reduced amount of data read during the backup process and fewer log records read during recovery, you can run backups more frequently. This feature is available for backups at both the database and the table space level.

Cumulative backups contain all data changed since the last full backup. Delta backups are non-cumulative, and contain only the data changed since the last full or incremental backup.

- **Backup to named pipe**

Named pipes will provide additional backup targets and sources for DB2 restores such as compression. The named pipe for backup and restore must be created on the same machine in the local file system before backing up the database.

- **Parallel roll forward recovery**

Multiple agents will work on recovery on an SMP machine. Log records distributed to these agents can be concurrently reapplied, improving crash and database roll forward recovery. DMS table space containers can also be created or resized in parallel.

Although some components of recovery cannot be done in parallel, most can. Table space roll forward recovery will not use multiple agents. The log records will be parallelized at page level, meaning that the same agent will process the log records for the same data page. The most common log records (insert, delete, add key, update, and delete key) can run in parallel.

- **Relocate Database**

The command db2relocatedb renames a database, or relocates a database or part of a database (for example, container, log directory) as specified in the configuration file provided by the user. This tool makes the necessary changes to the DB2 instance and database support files.

To change the name of the database TESTDB to PRODDB in the instance DB2INST1 that resides on the path /home/db2inst1, create the following configuration file:

```
DB_NAME=TESTDB,PRODDB
DB_PATH=/home/db2inst1
INSTANCE=db2inst1
NODENUM=0
```

Save the configuration file as relocate.cfg and use the following command to make the changes to the database files:

```
db2relocatedb -f relocate.cfg
```

Example 1

To move the database DATAB1 from the instance JSMITH on the path /dbpath to the instance PRODINST, do the following:

1. Move the files in the directory /dbpath/jsmith to /dbpath/prodinst.
2. Use the following configuration file with the db2relocatedb command to make the changes to the database files:

```
DB_NAME=DATAB1
DB_PATH=/dbpath
INSTANCE=jsmith,prodinst
NODENUM=0
```

Example 2

The database PRODDB exists in the instance INST1 on the path /databases/PRODDB. The location of two table space containers needs to be changed as follows:

- SMS container /data/SMS1 needs to be moved to /DATA/NewSMS1.
- DMS container /data/DMS1 needs to be moved to /DATA/DMS1.

After the physical directories and files have been moved to the new locations, the following configuration file can be used with the db2relocatedb command to make changes to the database files so that they recognize the new locations:

```
DB_NAME=PRODDB
DB_PATH=/databases/PRODDB
INSTANCE=inst1
NODENUM=0
CONT_PATH=/data/SMS1,/DATA/NewSMS1
CONT_PATH=/data/DMS1,/DATA/DMS1
```

Example 3

The database TESTDB exists in the instance DB2INST1 and was created on the path /databases/TESTDB. Table spaces were then created with the following containers:

```
TS1
TS2_Cont0
TS2_Cont1
/databases/TESTDB/TS3_Cont0
/databases/TESTDB/TS4/Cont0
/Data/TS5_Cont0
/dev/rTS5_Cont1
```

TESTDB is to be moved to a new system. The instance on the new system will be NEWINST and the location of the database will be /DB2. When moving the database, all of the files that exist in the /databases/TESTDB/db2inst1 directory must be

moved to the /DB2/newinst directory. This means that the first five containers will be relocated as part of this move. (The first three are relative to the database directory, and the next two are relative to the database path.) Since these containers are located within the database directory or database path, they do not need to be listed in the configuration file. If the two remaining containers are to be moved to different locations on the new system, they must be listed in the configuration file. After the physical directories and files have been moved to their new locations, the following configuration file can be used with db2relocatedb to make changes to the database files so that they recognize the new locations:

```
DB_NAME=TESTDB
DB_PATH=/databases/TESTDB,/DB2
INSTANCE=db2inst1,newinst
NODENUM=0
CONT_PATH=/Data/TS5_Cont0,/DB2/TESTDB/TS5_Cont0
CONT_PATH=/dev/rTS5_Cont1,/dev/rTESTDB_TS5_Cont1
```

Example 4

The database TESTDB has two partitions on nodes 10 and 20. The instance is SERVINST and the database path is /home/servinst on both nodes. The name of the database is being changed to SERVDB and the database path is being changed to /databases on both nodes. In addition, the log directory is being changed on node 20 from /testdb_logdir to /servdb_logdir. Since changes are being made to both nodes, a configuration file must be created for each node, and db2relocatedb must be run on each node with the corresponding configuration file. On node 10, the following configuration file will be used:

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=10
```

On node 20, the following configuration file will be used:

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=20
LOG_DIR=/testdb_logdir,/servdb_logdir
```

In a partitioned database environment, this tool must be run against every partition that requires changes. A separate configuration file must be supplied for each partition, including the DBPARTITIONNUM value of the partition being changed. For example, if the name of a database is being changed, every partition will be affected, and the db2relocatedb command must be run with a separate configuration file on each partition. If containers belonging to a single partition are being moved, the db2relocatedb command only needs to be run once on that partition.

DB2 Advanced Recovery and HA Workshop



CF49 — DB2 Advanced Recovery Workshop (4 days with Labs)

- Recovery log management / User exit support
- Dropped table recovery
- Table space recovery
- Incremental backup and restore
- Database crash recovery
- Redirected restore
- Additional recovery facilities
- Recovery history file
- Disaster recovery of DB2 databases
- DB2 high availability and split mirror support
- DB2 partitioned database recovery considerations
- DB2 high availability disaster recovery (HADR)

© Copyright IBM Corporation 2007

Figure 7-37. DB2 Advanced Recovery and HA Workshop

CF238.3

Notes:

You, or other people in your organization, may desire additional details. Please visit www.ibm.com/services/learning.

DB2 Advanced Recovery (CF49)

Overview

Gain a deeper understanding of the advanced recovery features of DB2 for Linux, UNIX, and Windows environments with multiple partition databases. Get practical experience in the planning and utilization of a wide variety of DB2 recovery facilities, in a series of database recovery scenarios you complete during lab exercises using DB2 Enterprise Server Edition.

Skills taught

- Gain a better understanding of the unique recovery planning requirements for DB2 ESE.
- Explore the DB2 recovery facilities and database configuration options.
- Plan the implementation of a user exit for archival of database logs.

- Recover a DB2 table following a DROP TABLE command issued in error.
- Plan and execute the recovery of tablespaces to a selected point in time.
- Effectively utilize incremental backup and restore to reduce the size and duration of DB2 database backups.
- Gain a better understanding of DB2 crash recovery facilities.
- Utilize the redirected restore option to recover DB2 data to alternate disk configurations and invoke the db2relocatedb command to alter the configuration of a DB2 database.
- Execute recovery scenarios, including loss of DB2 log data using the DB2 log mirroring option.
- Utilize the information in the DB2 recovery history file to plan and execute various DB2 utilities.
- Understand the benefits and limitations of disaster recovery options for DB2 databases.
- Explore the options for operation of DB2 databases in high availability environments including the use of split mirrors of the database.
- **HADR High Availability and Disaster Recovery**

Checkpoint

Exercise — Unit Checkpoint

___ 1. State the three methods of recovery.

___ 2. What is required to avoid the need for a roll forward?

___ 3. How might a database be placed in roll forward pending status?

___ 4. How might a table space be placed in roll forward pending status?

a table space?

Unit summary

Having completed this unit, you should be able to:

- Describe the major principles and methods for backup and recovery
- State the three types of recovery used by DB2
- Explain the importance of logging for backup and recovery
- Describe how data logging takes place, including circular logging and archival logging
- Use the BACKUP, RESTORE, and ROLLFORWARD commands
- Perform a table space backup and recovery
- Restore a database to the end-of-logs or to a point-in-time
- Discuss the configuration parameters and the recovery history file and use these to handle various backup and recovery scenarios

© Copyright IBM Corporation 2007

Figure 7-38. Unit summary

CF238.3

Notes:

Unit 8. Locking and concurrency

What this unit is about

This unit addresses the concepts of the DB2 database manager's implementation of locking to ensure data integrity while permitting multiple applications access to data. The database administrator needs a working knowledge of locking administration to successfully manage factors that influence the locking strategies used. Using this knowledge, the administrator can tailor the environment to meet the installation's concurrency requirements.

What you should be able to do

After completing this unit, you should be able to:

- Explain why locking is needed
- List objects that can be locked
- Describe and discuss the various lock modes and their compatibility
- Explain four different levels of data protection
- Set isolation level and lock time out for current activity
- Explain lock conversion and escalation
- Describe the situation that causes deadlocks

How you will check your progress

Accountability:

- Checkpoint
- Lab exercises

References

Administration Guide

System Monitor Guide and Reference

Unit objectives



After completing this unit, you should be able to:

- Explain why locking is needed
- List objects that can be locked
- Describe and discuss the various lock modes and their compatibility
- Explain four different levels of data protection
- Set isolation level and lock time out for current activity
- Explain lock conversion and escalation
- Describe the situation that causes deadlocks

© Copyright IBM Corporation 2007

Figure 8-1. Unit objectives

CF238.3

Notes:

8.1 Locking objects, strategies, and modes

Why are locks needed?

- Ensure data integrity while permitting multiple applications to access data
- Prohibit applications from accessing uncommitted data written by other applications (*unless Uncommitted Read isolation level is used*)
- Control undesirable effects
 - Lost data due to concurrent updates
 - Unrepeatable reads
 - Phantom read phenomenon



© Copyright IBM Corporation 2007

Figure 8-2. Why are locks needed?

CF238.3

Notes:

Because many users access and change data in a relational database, the database manager must be able both to allow users to make these changes and to ensure that data integrity is preserved. *Concurrency* refers to the sharing of resources by multiple interactive users or application programs at the same time.

The primary reasons why locks are needed are:

- **Ensure data integrity.** Stop one application from accessing or changing a record while another application has the record locked for its use.
- **Access to uncommitted data.** Application A might update a value in the database, and application B might read that value before it was committed. If the value of A is not later committed, but backed out, calculations performed by B are based on uncommitted (and presumably invalid) data. Of course, you may want to read even uncommitted data, for example to get a rough count of the number of records of a particular type without the guarantee of instantaneous precision. You can use an Uncommitted Read (UR) isolation level to do this — we will see more about this later.

The database manager controls this access to prevent undesirable effects, such as:

- **Lost updates.** Two applications, A and B, might both read the same row from the database and both calculate new values for one of its columns based on the data these applications read. If A updates the row with its new value and B then also updates the row, the update performed by A is lost.
- **Nonrepeatable reads.** Some applications involve the following sequence of events: application A reads a row from the database, then goes on to process other SQL requests. In the meantime, application B either modifies or deletes the row and commits the change. Later, if application A attempts to read the original row again, it receives the modified row or discovers that the original row has been deleted.
- **Phantom Read Phenomenon.** The phantom read phenomenon occurs when:
 1. Your application executes a query that reads a set of rows based on some search criterion.
 2. Another application inserts new data or updates existing data that would satisfy your application's query.
 3. Your application repeats the query from Step 1 (within the same unit of work).
 4. Some additional (phantom) rows are returned as part of the result set, but were not returned when the query was initially executed (Step 1).

Classification of locks



Locks can be categorized based on the following attributes:

- **OBJECT** — what resource is locked?
- **MODE** — what type of lock is being used?
- **DURATION** — when will the lock be released?

and four different levels of protection to isolate data:

- Uncommitted read (UR)
- Cursor stability (CS)
- Read stability (RS)
- Repeatable read (RR)

Isolation levels will be discussed later in this unit

© Copyright IBM Corporation 2007

Figure 8-3. Classification of locks

CF238.3

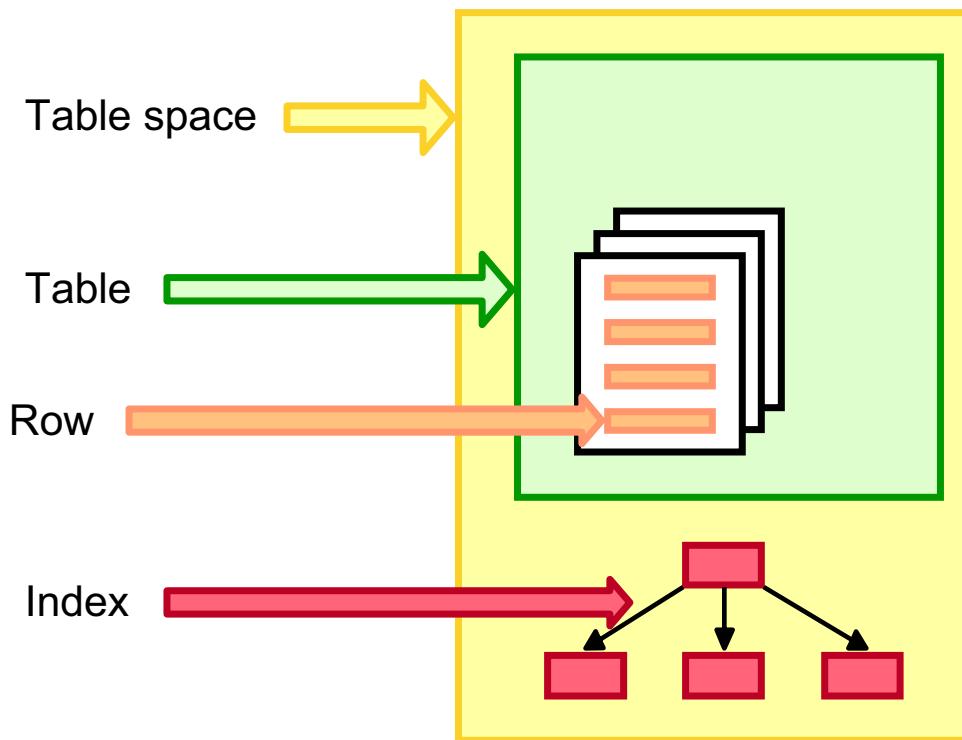
Notes:

Our discussion of locks is divided into a number of subtopics:

- **Object** — What resource is locked? — databases, table spaces, tables, rows
- **Mode** — What type of lock is being used? — we will look at the variety of table and row locks in detail, and how they interact
- **Duration** — When is the lock gained? and when will it be released?

Isolation levels are discussed in greater detail later in this unit as a separate topic.

Objects of locks



© Copyright IBM Corporation 2007

Figure 8-4. Objects of locks

CF238.3

Notes:

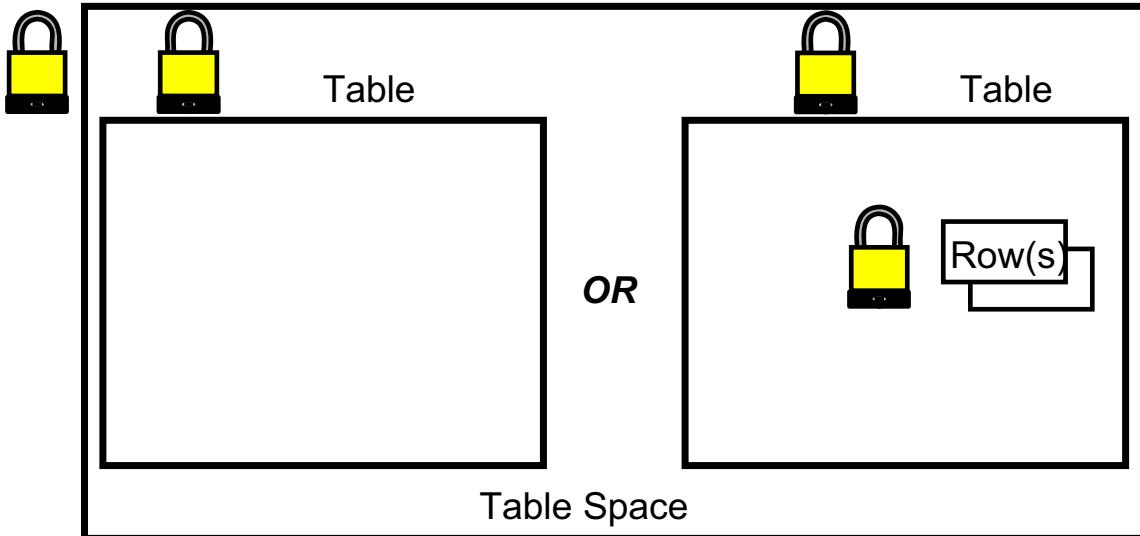
DB2 utilizes locks on table spaces, tables, rows, and indexes to ensure data integrity while providing application concurrency.

Indexes and table space locks are shown on this visual in order to identify all objects that are subject to locking. However, the locks used on indexes and table spaces are not generally examined during problem determination for application processing. Objects that are the focus of such analysis include the table and row.

For example, a table space lock may be obtained to support a QUIESCE request. However, such a request is not considered typical application processing.

This topic will focus on table and row locking.

Locking strategies



DB2 employs *either* strict table locking **OR** table locking in conjunction with row locking for typical application processing

© Copyright IBM Corporation 2007

Figure 8-5. Locking strategies

CF238.3

Notes:

The above visual illustrates the locking strategies that DB2 utilizes for access to data.

The table space lock, if required, is acquired *first* in either strategy. It is shown on the visual for completeness, but will not generally be of concern when analyzing application concurrency due to the nature of the lock mode.

DB2 may lock only the table (sometimes called *strict table locking*), or may lock the table and a row or many rows.

- If the locking strategy is only on the table, then all rows in the table will be affected to the same degree.
- If the locking strategy is on the table and the underlying rows, then the entire table is locked with a lock that allows a great deal of concurrency at the table level, followed by locking of requested rows.

**Note**

The mode of the lock (whether it will support READ or CHANGE) has been **intentionally** omitted from the discussion at this point. It will be addressed later in this unit.

Table lock modes



IN	Intent None
IS	Intent Share
IX	Intent eXclusive
SIX	Share with Intent eXclusive
S	Share
U	Update
X	eXclusive
Z	superexclusive

Row Locking also used

Strict Table Locking

(See next page)

© Copyright IBM Corporation 2007

Figure 8-6. Table lock modes

CF238.3

Notes:

The lock modes listed above are used by DB2 at the table level and are defined below:

- **IN — Intent None:** The lock owner can read any data in the table including uncommitted data, but cannot update any of it. Other concurrent applications can read or update the table. No row locks are acquired by the lock owner. Both table spaces and tables can be locked in this mode.
- **IS — Intent Share:** The lock owner can read any data in the locked table if an S lock can be obtained on the target rows. The lock owner cannot update the data in the table. Other applications can read or update the table, as long as they are not updating rows on which the lock owner has an S lock. Both table spaces and tables can be locked in this mode.
- **IX — Intent Exclusive:** The lock owner can read and update data provided that an X lock can be obtained on rows to be changed, and that a U or S lock can be obtained on rows to be read. Other concurrent applications can both read and update the table, as long as they are not reading or updating rows on which the lock owner has an X lock. Both table spaces and tables can be locked in this mode.

- **SIX — Share with Intent Exclusive:** The lock owner can read any data in the table and change rows in the table provided that it can obtain an X lock on the target rows for change. Row locks are not obtained for reading. Other concurrent applications can read the table. Only a table object can be locked in this mode. The SIX table lock is a special case. It is obtained if an application possesses an IX lock on a table and requests an S lock, or vice versa. The result of lock conversion in these cases is the SIX lock.
- **S — Share:** The lock owner and all concurrent applications can read but not update any data in the table and will not obtain row locks. Tables can be locked in this mode.
- **U — Update:** The lock owner can read any data in the table and may change data if an X lock on the table can be obtained. No row locks are obtained. This type of lock may be obtained if an application issues a SELECT... 'for update'. Other units of work can read the data in the locked object, but cannot attempt to update it. Tables can be locked in this mode.
- **X — Exclusive:** The lock owner can read or update any data in the table. Row locks are not obtained. Only uncommitted read applications can access the locked object. Tables can be locked in this mode.
- **Z — Super Exclusive:** This lock is acquired on a table in certain conditions, such as when the table is altered or dropped, or for some types of table reorganization. No other concurrent application can read or update the table. Tables and table spaces can be locked in this mode. No row locks are obtained.

The modes **IS**, **IX**, and **SIX** are used at the table level to **SUPPORT** row locks. They permit row-level locking while preventing more exclusive locks on the table by other applications.

The following examples are used to further clarify the lock modes of **IS**, **IX**, and **SIX**:

- An application obtains an IS lock on a table. That application may acquire a lock on a row for read only. Other applications can also **READ** the same row. In addition, other applications can **CHANGE** data on other rows in the table.
- An application obtains an IX lock on a table. That application may acquire a lock on a row for change. Other applications can **READ/CHANGE** data on other* rows in the table.
- An application obtains an SIX lock on a table. That application may acquire a lock on a row for change. Other applications can **ONLY READ** other* rows in the table.

The modes of **S**, **U**, **X**, and **Z** are used at the table level to enforce the strict table locking strategy. **No row-level locking** is used by applications that possess one of these modes.

The following examples are used to further clarify the lock modes of **S**, **U**, **X**, and **Z**:

- An application obtains an S lock on a table. That application can read any data in that table. It will allow other applications to obtain locks that support read-only requests for any data in the entire table. No application can **CHANGE** any data in the table until the S lock is released.
- An application obtains a U lock on a table. That application can read any data in that table, and may eventually change data in that table by obtaining an X lock. Other applications can only **READ** data in the table.

- An application obtains an X lock on a table. That application can read and change any or all of the data in the table. No other application can access data in the entire table for **READ* or CHANGE**.
- An application obtains a Z lock on a table. That application can read and change any or all of the data in the table. No other application can access data in the entire table for **READ or CHANGE**.

The mode of **IN** is used at the table to permit the concept of Uncommitted Read. An application using this lock will **not** obtain row-level locks.

* Denotes an exception to a given application scenario. Applications that use Uncommitted Read can read rows that have been changed. More details regarding Uncommitted Read are provided later in this unit.



Note

Some of the lock modes discussed are also available at the table space level. For example, an IS lock at the table space level supports an IS or S lock at the table level. However, further details regarding table space locking are not the focus of this unit.

Row lock modes



Row Lock		Minimum* Supporting Table Lock
S	Share	IS
U	Update	IX
X	eXclusive	IX
W	Weak exclusive	IX
NS	Next key Share	IS
NW	Next key Weak exclusive	IX

An application does not acquire

Row locks

if it is using Table Locks of

S, U, X, or Z

© Copyright IBM Corporation 2007

Figure 8-7. Row lock modes

CF238.3

Notes:

The above modes are for row locks. The definitions are similar to the definitions for corresponding table locks, except that the object of the lock is a row.

- **S—Share:** The row is being READ by one application and is available for READ ONLY by other applications.
- **U—Update:** The row is being READ by one application but is possibly to be changed by that application. The row is available for READ ONLY by other applications. The major difference between the U lock and the S lock is the INTENT TO UPDATE. The U lock will support cursors that are opened with the **FOR UPDATE OF** clause. Only one application can possess a U lock on a row.
- **X—Exclusive:** The row is being changed by one application and is not available for other applications, except those that permit Uncommitted Read.
- **W—Weak Exclusive:** This lock is acquired on the row when a row is inserted into a non-catalog table and a duplicate key for a unique index is encountered. The lock owner can change the locked row. This lock is similar to an X lock except that it is compatible with the NW lock.

- **NS — Next Key Share:** The lock owner and all concurrent applications can read, but not change, the locked row. Only individual rows can be locked in NS mode. This lock is acquired in place of a share (S) lock on data that is read with the RS or CS isolation levels.
- **NW — Next Key Weak Exclusive:** This lock is acquired on the next row when a row is inserted into the index of a non-catalog table. The lock owner can read, but not change, the locked row. This is similar to X and NX locks, except that it is compatible with the W and NS locks.

Row locks are only requested by applications that have supporting locks at the table level. These supporting locks are the INTENT locks: IS, IX, and SIX.

* Denotes the least restrictive lock necessary. However, this does not imply that the table lock listed is the only table lock that supports the row lock listed. For example, an application that possesses an IX table lock could possess S, U, or X locks on rows. Likewise, an application that possesses a SIX table lock could possess X locks on rows.

Lock mode compatibility

MODE OF LOCK A	MODE OF LOCK B							
	IN	IS	S	IX	SIX	U	X	Z
IN	YES	YES	YES	YES	YES	YES	YES	NO
IS	YES	YES	YES	YES	YES	YES	NO	NO
S	YES	YES	YES	NO	NO	YES	NO	NO
IX	YES	YES	NO	YES	NO	NO	NO	NO
SIX	YES	YES	NO	NO	NO	NO	NO	NO
U	YES	YES	YES	NO	NO	NO	NO	NO
X	YES	NO	NO	NO	NO	NO	NO	NO
Z	NO	NO	NO	NO	NO	NO	NO	NO

Table Locks

Row Locks

LOCK A MODE	MODE OF LOCK B					
	S	U	X	W	NS	NW
S	YES	YES	NO	NO	YES	NO
U	YES	NO	NO	NO	YES	NO
X	NO	NO	NO	NO	NO	NO
W	NO	NO	NO	NO	NO	YES
NS	YES	YES	NO	NO	YES	YES
NW	NO	NO	NO	YES	YES	NO

© Copyright IBM Corporation 2007

Figure 8-8. Lock mode compatibility

CF238.3

Notes:

The symbols **A** and **B** in the above diagrams are used to represent two different applications. The chart regarding table locks can be used to determine if the two applications can run concurrently if they are requesting access to the same table with a given lock mode.

For example, if application A obtains an IS lock against a given table, application B could obtain an IN, IS, S, IX, SIX, or U lock against the same table at the same time. However, an X or Z lock would not be permitted at the same time.

This particular example illustrates the concept of the IS lock acting as a supporting lock for a lower level of locking. The only table locks that are not compatible are the X and Z locks, which would require exclusive use of the table. The presence of the IS lock indicates that a lower level of locking is required for this table, and the X or Z lock request is not given.

Study of the chart simply reinforces the definitions of table and row lock modes presented on the previous two pages. Review the row for IX under application A. Assume that application A obtains an IX lock on the table Y. This lock indicates that the application intends to obtain locks to support change at the row level. The application will allow other

rows to be read and updated, but will prevent access to the *target* rows (with the exception of Uncommitted Read applications.) Examine each of the possible *competing* table locks that application B might request:

- **IN** — No row lock intention. This lock is compatible. There will be no contention since application B is requesting Uncommitted Read. Even rows changed and not committed by application A are available. (The Z lock is the only mode that is not compatible with IN.)
- **IS** — Intent to lock for read only at the row level. This lock is compatible. There may be contention at the row level if application A is changing the same row that application B wants to read. The *Row Locks* table would need to be examined: if application A has acquired an X or a W lock on the row that application B is attempting to read, then application B will need to wait. Otherwise, the two applications can proceed with concurrency.
- **S** — Share lock at the table level. This lock is NOT compatible, since the S lock states that the entire table is available for READ ONLY by the application possessing the lock and all other applications. The IX lock states an intent to change data at the row level, which contradicts the requirement for READ ONLY. Therefore, application B could not obtain the S lock.
- **IX** — Intent to lock for change at the row level. This lock is compatible. There may be contention at the row level if application A is changing the same row that application B wants to change. The *Row Locks* table would need to be examined: if application A has acquired an X or a W lock on the row that application B is attempting to change, then application B will need to wait. Otherwise, the two applications can proceed with concurrency.
- **SIX** — The SIX lock states that a lock request for changing data may be required at the row level for the application possessing the lock. In addition, the rest of the table is available for READ ONLY applications. The IX lock implies change at the row level as well. Application B could not obtain the SIX lock on the table because of the *S characteristic* of the SIX lock, which is not compatible with the IX lock already assumed owned by application A.
- **U** — Read with intent to update. This table level lock states that the application possessing the lock may read any data, and may potentially exchange the U lock for an X lock. However, until this exchange is done, other applications can obtain locks supporting READ ONLY. Application B would NOT be able to obtain the U lock at the same time that application A possessed an IX lock on the same table.
- **X** — The application possessing this mode of lock on the table requires exclusive use of the table. No other access, with the exception of Uncommitted Read applications, is permitted. The IX lock possessed by application A would prevent application B from obtaining an X lock.
- **Z** — The application possessing this mode of lock excludes all other access to the table, including Uncommitted Read applications. Since application A has obtained an incompatible lock (IX), application B would not be able to obtain the Z lock at the same time.

The same type of statements could be logically derived for the other rows in the chart.

Many different applications could have compatible locks on the same object. For example, ten transactions may have IS locks on a table, and five different transactions may have IX locks on the same table. There is no concurrency problem at the table level in such a scenario. However, there may be lock contention at the row level:

- The basic concept of the row lock matrix is that rows being READ by an application can be READ by other applications, and that rows being changed by an application are not available to other applications that use row locking.
- Note that the U row lock is not compatible with another U row lock. Only one application can read a row with the INTENT TO UPDATE. This U lock reduces the number of deadlocks that occur when applications perform updates and deletes via cursors. When a row is Fetched using a cursor declared ...**FOR UPDATE OF...**, the U row lock is used.

8.2 Isolation levels

Introduction to isolation levels



- DB2 provides different levels of protection to isolate data:
 - Uncommitted read (UR)
 - Cursor stability (CS) - default
 - Read stability (RS)
 - Repeatable read (RR)
- Cursor stability is the default isolation level
- Isolation level can be specified for a session, a client connection, or an application before a database connection
 - For embedded SQL, the level is set at bind time
 - For dynamic SQL, the level is set at run time

© Copyright IBM Corporation 2007

Figure 8-9. Introduction to isolation levels

CF238.3

Notes:

This visual sets the stage for our discussion on isolation levels.

The isolation level chosen for an application can impact both the lock strategy and the duration of row locks. The isolation level can be specified during program preparation or bind.

Cursor stability is the default.

Isolation levels

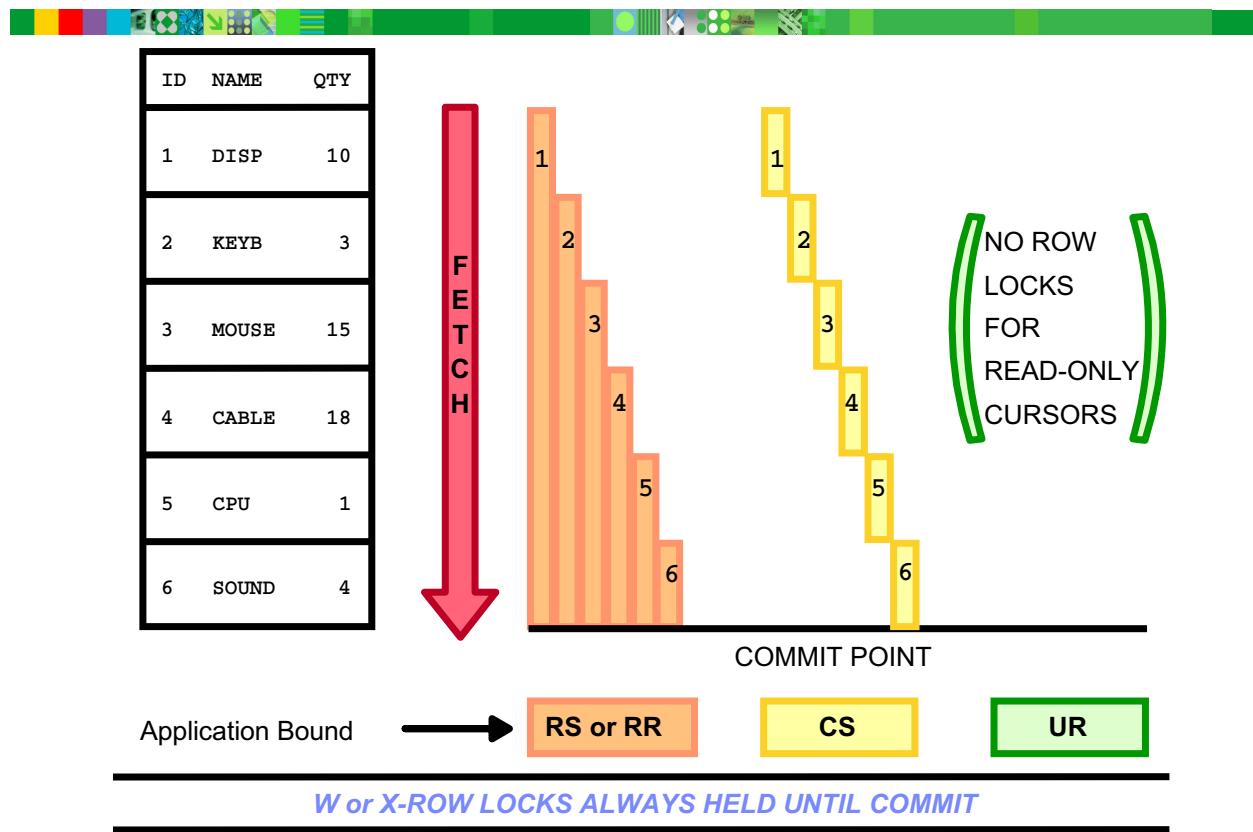


Figure 8-10. Isolation levels

CF238.3

Notes:

This visual details the impact of isolation level on row lock duration. For the following notes, assume that row locks are being obtained:

- **Read Stability (RS):** NS row locks are held until the next commit or rollback. If a row is read multiple times within a unit of work, the value for the row will not change. However, the result set can change if a cursor that has been processed is subsequently reopened with the same search criteria within an unit of work. This is because Read Stability does not prevent new rows from being *added* to the result set via insert or update activity.

This isolation level reduces concurrency at the row level, but may be useful for applications that require examination of multiple rows before a processing decision can be made.

- **Repeatable Read (RR):** S row locks are held until the next commit or rollback. If a row is read multiple times within a unit of work, the value for the row will not change. Also, the result set will not change if a cursor that has been processed is subsequently reopened using the same search criteria within a unit of work.

This isolation level reduces concurrency at the row level, but may be useful for applications that require examination of multiple rows before a processing decision can be made and also requires the result set to be static if the same criteria is used to open a cursor a second time within a unit of work.

- **Cursor Stability (CS):** NS row locks are held only while the cursor is positioned on the row. Rows that have been previously examined within a unit of work are not locked.

This isolation level increases concurrency at the row level, and it is normally preferred for most application requirements.

- **Uncommitted Read (UR):** S row locks are not obtained when single rows are read or when READ-ONLY cursors are OPENED and FETCHED. A cursor is READ-ONLY if:

- The result table for the SELECT statement is read-only. For example, the SELECT statement contains an ORDER BY clause. (Other syntax constructs that cause a result table to be read-only are documented in the *SQL Reference*.)
- The cursor is declared with FOR FETCH ONLY.

For non read-only cursors, the locking behavior is the same as cursor stability.

This isolation level may improve performance of applications since waits for S row locks are eliminated. However, the application programmer must be aware of any data integrity risks associated with this isolation level.

Isolation level applies only to rows that are read. Rows that are changed require an X or a W lock to be acquired by the application. **X or W locks are not released until rollback or commit, regardless of the application's isolation level.**

If a temporary table is needed to support an application using cursor stability, the S row locks are obtained and released during the OPEN of the cursor. FETCH from the temporary table may therefore present a row to the application that has since been changed or deleted in the underlying base table. Temporary tables may be used for read-only cursors that require sort activity. The use of temporary tables is externalized via the EXPLAIN function.



Note

SET CURRENT ISOLATION

To change the isolation level for a specific connection session, you can update a special registry variable. This is covered in an upcoming visual.

Repeatable Read versus Read Stability



- Repeatable Read (RR)

- Locks all the rows an application references within a unit of work — *every row referenced is locked*
- If the application issues the same query (SELECT) more than once within a unit of work, the same result set is obtained each time
- No other application can UPDATE, DELETE, or INSERT a row that would affect the result table until the unit of work complete

- Read Stability (RS)

- Locks all the rows an application retries within a unit of work — *only rows that qualify are locked*
- If the application issues the same query more than once within a unit of work, additional *phantom rows* may be returned

© Copyright IBM Corporation 2007

Figure 8-11. Repeatable Read versus Read Stability

CF238.3

Notes:

RR and RS both lock all the rows an application retrieves within a unit of work.

Using repeatable read, a SELECT statement issued by an application more than once within a unit of work gives the same result each time. No other application can update, delete, or insert a row that would affect the result set until the unit of work completes. Every row that is referenced is locked, not just the rows that are retrieved, preventing *phantom rows* from occurring. Thus if you scan 10,000 rows and apply predicates to them, locks are held on all 10,000 rows, even though only 10 rows may qualify.

Using read stability, if your application issues the same query more than once, you may see additional *phantom rows*. A phantom row can occur in the following example situation:

1. Application 1 reads the set of n rows that satisfy a search condition.
2. Application 2 then inserts one or more rows that satisfy the search condition.
3. Application 1 reads the set of rows again with the same search condition, and obtains both the original rows and the rows inserted by application 2.

RS locks only the rows that qualify — perhaps just 10 out of the 10,000 rows in the table.

Summary of isolation levels



Isolation level	Access to uncommitted data	Nonrepeatable reads	Phantom read phenomenon
Repeatable Read (RR)	Not possible	Not possible	Not possible
Read Stability (RS)	Not possible	Not possible	Possible
Cursor Stability (CS)	Not possible	Possible	Possible
Uncommitted Read (UR)	Possible	Possible	Possible

© Copyright IBM Corporation 2007

Figure 8-12. Summary of isolation levels

CF238.3

Notes:

The table summarizes the different isolation levels in terms of their impact on undesirable effects. Note that all of the isolation levels protect against lost data due to concurrent updates.

SET CURRENT ISOLATION



- Assigns a value to the CURRENT ISOLATION special register

```
SET [CURRENT] ISOLATION [=] {UR | CS | RR | RS | RESET}
```

- CURRENT ISOLATION can be set anytime while connected to a database (special registers are attributes of the connection session)
- Applies to the current session only
- The register setting is used immediately for subsequent SQL statements *until value is set again*, thus allowing different isolation to be applied to different SELECT (and other) statements even within one UOW/transaction
- You can read the current register setting with:

```
VALUES CURRENT ISOLATION
```

© Copyright IBM Corporation 2007

Figure 8-13. SET CURRENT ISOLATION

CF238.3

Notes:

The SET CURRENT ISOLATION statement assigns a value to a special registry. This statement can be embedded in an application program or issued through the use of dynamic SQL statements. The registry setting is used immediately for following SQL statements until the value is set again, thus allowing, for example, different isolation to be applied to different SELECT statements, even within one Unit of Work.

The SET CURRENT ISOLATION statement is not under transaction control. The register setting is retained across COMMIT boundaries.

This statement can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can be dynamically prepared. No particular authorization is required to execute this statement.

The syntax of the statement (as shown in the visual above) is:

```
SET [CURRENT] ISOLATION [=] {UR | CS | RR | RS | RESET }
```

The value of the CURRENT ISOLATION special register is replaced by the specified value or set to blanks if RESET is specified.

The following syntax is also supported:

- The word CURRENT is optional *
- The equals sign is optional — but TO can be specified in place of the equal sign (=) *
- DIRTY READ can be specified in place of UR *
- READ UNCOMMITTED can be specified in place of UR *
- READ COMMITTED is recognized and upgraded to CS *
- CURSOR STABILITY can be specified in place of CS *
- REPEATABLE READ can be specified in place of RR *
- SERIALIZABLE can be specified in place of RR

The asterisk items (*) are for compatibility with IBM Informix SQL syntax.

8.3 Explicit versus implicit locking

Explicit versus implicit locking

- Locking is controlled by isolation level
- By default, DB2 uses row-level locking
- Database, table spaces, and tables can be explicitly locked
- Database, tables, and rows can be implicitly locked

© Copyright IBM Corporation 2007

Figure 8-14. Explicit versus implicit locking

CF238.3

Notes:

The lock mode that is used by an application is determined by the isolation level. Locks are placed on the table or on individual rows, or on *pages* of rows in the table. DB2 for Linux, UNIX, and Windows does not support page level locking. The default used by DB2 is row locking.

Databases, table spaces, and tables can be *explicitly* locked. Here are some examples of commands that can be used to explicitly lock these database objects:

- Database lock

```
CONNECT TO database IN EXCLUSIVE MODE
```

- Table space lock

```
QUIESCE table-spaces FOR TABLE table-name INTENT FOR UPDATE
```

- Table lock

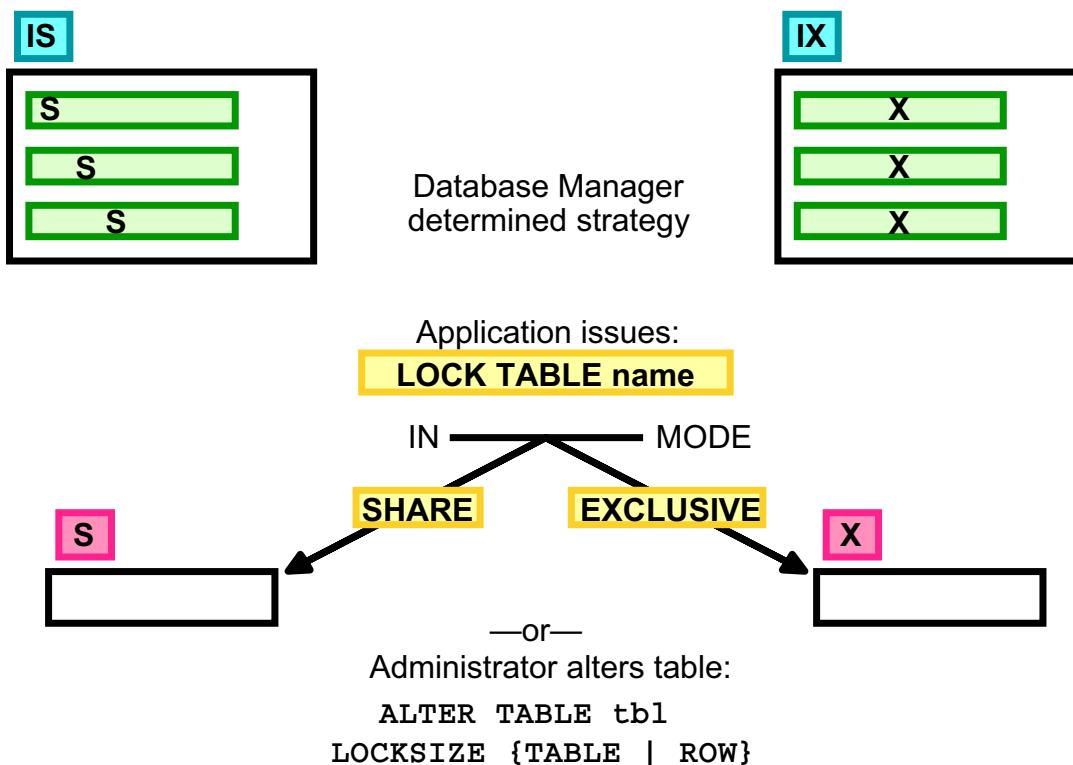
```
LOCK TABLE table-name IN EXCLUSIVE MODE
```

The LOCK TABLE statement is the subject of the next visual.

Database, tables, and rows can be *implicitly* locked. For example:

- Database is locked during full database restore
- Tables are locked during lock escalation
- Rows are locked through normal data modification (SELECT, INSERT, UPDATE, and DELETE)

LOCK TABLE statement



© Copyright IBM Corporation 2007

Figure 8-15. LOCK TABLE statement

CF238.3

Notes:

Isolation level and access strategy are factors that affect the database manager when it determines the locking strategy to use when reading or manipulating data. **Generally**, intent locks at the table level, and row locking, are used to support transaction-oriented applications.

However, the use of intent locking may not be appropriate for a given application.

The LOCK TABLE statement provides the application programmer with the flexibility to lock a table at a more restrictive mode than requested by the database manager. Only applications with a need for a more restrictive mode of lock should issue the LOCK TABLE statement. Such applications could include report programs that must show *snapshots* of the data at a given point in time, or data modifying programs that normally don't make changes to significant portions of a table except during certain periods such as for month-end processing.

SHARE MODE allows other processes to SELECT data in the TABLE, but does not allow INSERT, UPDATE, or DELETE operations.

EXCLUSIVE MODE prevents any other processes from performing any operation on the table, with the exception of Uncommitted Read applications.

Locks obtained via the LOCK TABLE statement are acquired when the statement is executed. These locks are released by commit or rollback.



Note

The visual is not intended to imply that an application can only request a more restrictive table lock of the same nature (IS to S / IX to X), although this would be the typical case.

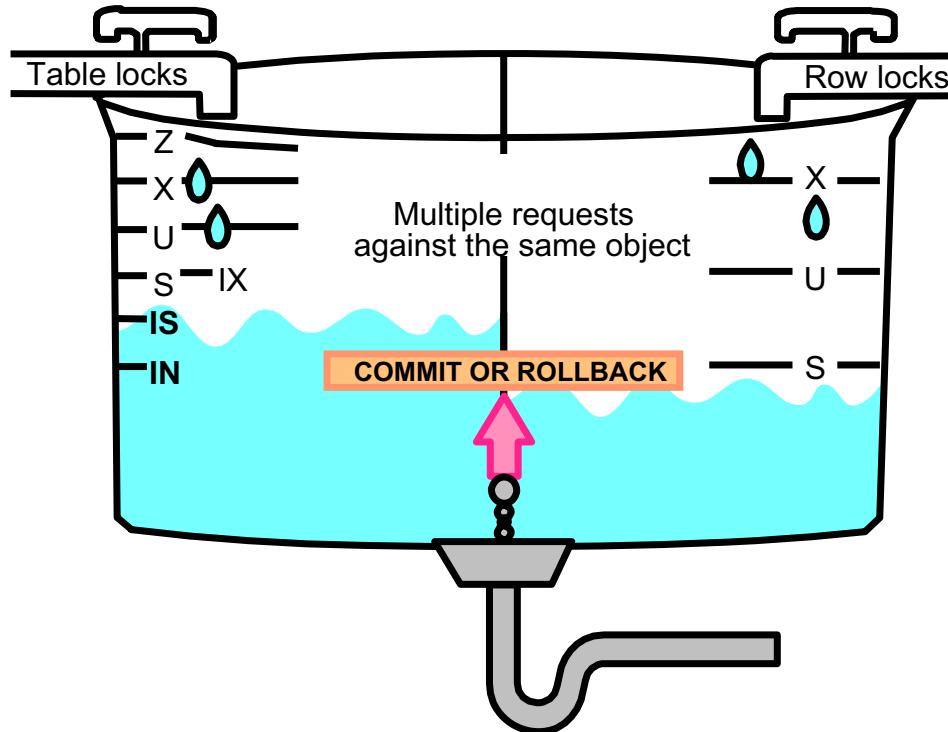
The table can also be altered to indicate the size (granularity) of locks used when the table is accessed. If LOCKSIZE TABLE is indicated, then the appropriate share or exclusive lock is acquired on the table, and intent locks (except intent none) are not used. Use of this value may improve the performance of queries by limiting the number of locks that need to be acquired. However, concurrency is also reduced since all locks are held over the complete table.

Even though the intent lock strategy is common for typical transaction-oriented applications, there are situations when strict table locking will be selected by the database manager. For example, the isolation level of Repeatable Read combined with an access strategy of TABLE SCAN or INDEX SCAN with no WHERE clause will be supported with a strict table lock. If the strict table locking that results causes unacceptable concurrency problems, the applications using Repeatable Read should be examined to determine if a different access strategy can be used or if the isolation level can be changed. Repeatable Read can be logically simulated, although the application code required to do so may carry a high cost for development, maintenance, or both.

Strict table locking cannot be avoided when issuing DDL against a table or index. When possible, the database administrator should restrict submission of such statements to periods of low activity.

In any case, strict table locks determined during the optimization process are externalized by the Explain function.

Lock conversion



© Copyright IBM Corporation 2007

Figure 8-16. Lock conversion

CF238.3

Notes:

Although the Explain tool provides information regarding the strategy determined at bind, the information provided is on a statement-by-statement basis. The overall locking requirement of an application on a table must be derived by combining knowledge of the application with the output of the Explain tool.

Lock conversion is the process of obtaining a lock with a greater degree of restrictiveness as SQL statements are processed within a unit of work that require different lock modes on the same object.

Lock modes convert to higher modes only and remain at the higher mode until commit or rollback. (This is not to imply that locks acquired under cursor stability isolation level will be held longer than previously described.) After this point, table and row locks requested in the next unit of work will establish a new *high water mark*.

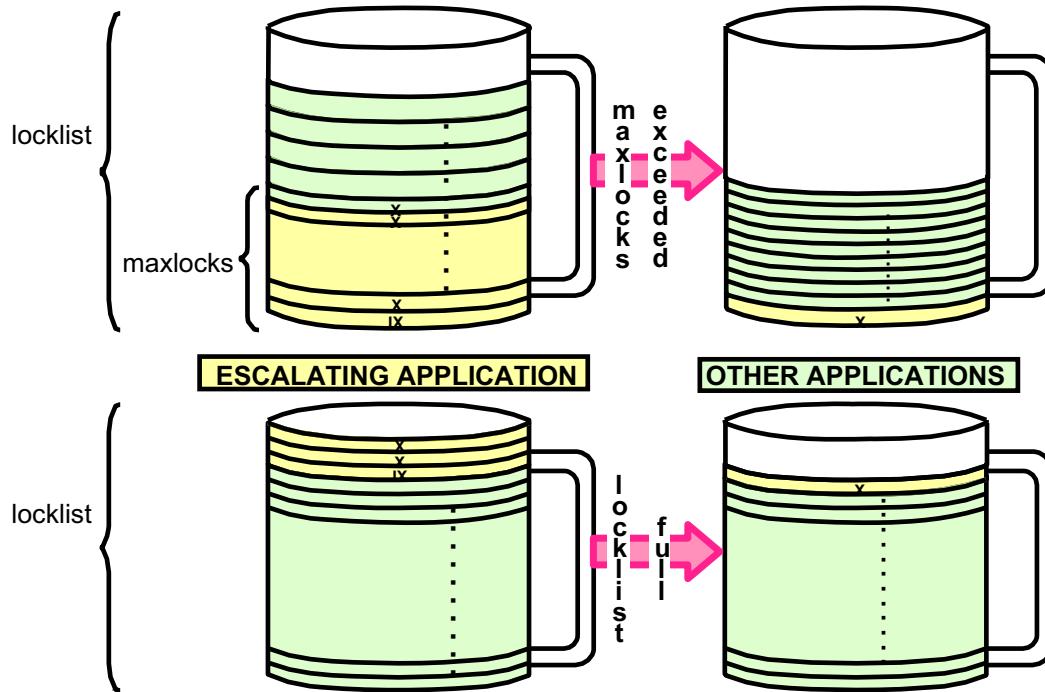
The SIX table lock is a special case. It is obtained if an application possesses an IX lock on a table and requests an S lock, or vice versa. The result of lock conversion in these cases is the SIX lock.

As an example of table lock conversion, consider the following application (assume RR for isolation and IXCOL = ? utilizes an INDEX SCAN):

Statement	Table lock required for statement (externally via Explain)	Conversion required?	Resultant lock
SELECT ... FROM A WHERE IXCOL = ?	IS	N/A	IS
SELECT ... FROM A	S	Yes	S
SELECT ... FROM A WHERE IXCOL = ?	IS	No	S
UPDATE A SET ... WHERE IXCOL = ?	IX	Yes	SIX
COMMIT	N/A	No	(locks released)
SELECT ... FROM A WHERE IXCOL = ?	IS	N/A	IS
LOCK TABLE A IN EXCLUSIVE MODE	X	Yes	X
UPDATE A SET ... WHERE IXCOL = ?	IX	No	X
ROLLBACK	N/A	No	(locks released)

All locks (row and table) are released at commit or rollback, with the exception of table locks associated with cursors declared WITH HOLD. These locks are held through commit.

Lock escalation



© Copyright IBM Corporation 2007

Figure 8-17. Lock escalation

CF238.3

Notes:

In order to service as many applications as possible, the database manager provides the function of lock escalation. This process entails obtaining a table lock and releasing row locks. The desired effect of the process is to reduce the overall storage requirement for locks by the database manager. This will enable other applications to obtain locks requested.

Two database configuration parameters have a direct impact on the process of lock escalation:

- **locklist** — The number of 4 KB pages allocated in the database global memory for lock storage. This parameter is configurable online.

The default value for locklist and maxlocks in DB2 9 is now AUTOMATIC, so the self tuning memory management routines can adjust the size of the locklist and maxlocks to match the demands of the current workload.

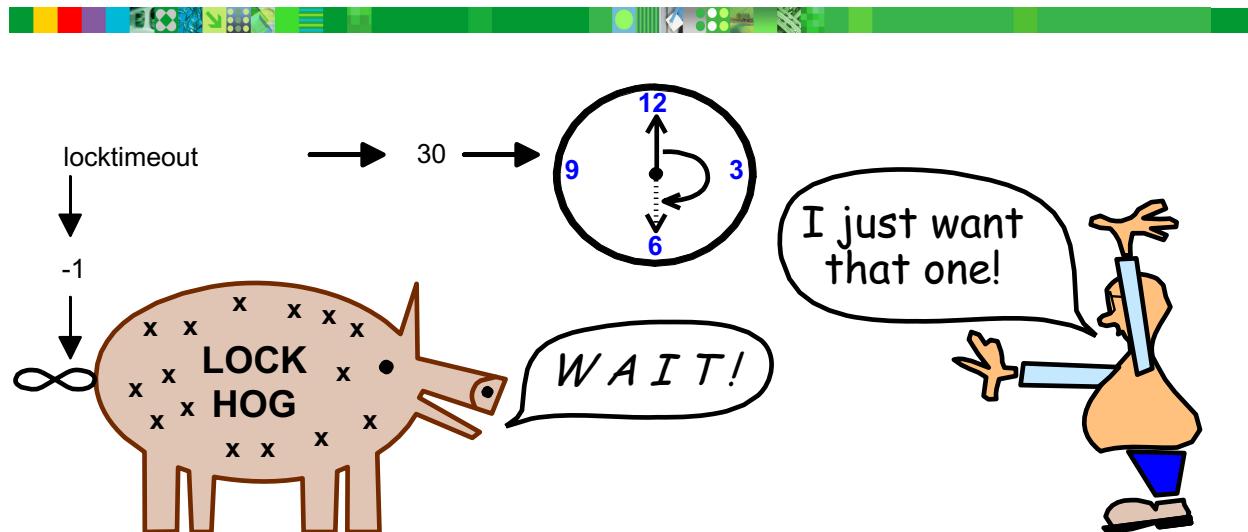
- **maxlocks** — The percentage of the total locklist permitted by a single application. This parameter is configurable online.

Lock escalation can occur in two different situations:

1. A single application requests a lock that will cause the application to exceed the percentage of the total locklist as defined by maxlocks. The database manager will attempt to free memory space by obtaining a table lock and releasing row locks for the requesting application.
 2. An application triggers lock escalation because the total locklist is full. The database manager will attempt to free memory space by obtaining a table lock and releasing row locks for the requesting application. Note that the application being escalated may or may not have a significant number of locks. The total lock volume may be reaching a threshold because of high system activity where no individual application has reached the limit established by maxlocks.
- A lock escalation attempt can fail. If a failure occurs, the application for which escalation has been attempted will receive a -912 SQLCODE. Such a return code should be handled by the application.

8.4 Lock timeouts and deadlock

Lock wait and timeout



If the application *hogging* the locks
doesn't COMMIT or ROLLBACK
other applications wait until lock is
available or timeout exceeded

© Copyright IBM Corporation 2007

Figure 8-18. Lock wait and timeout

CF238.3

Notes:

Applications that request a lock that is not compatible with the existing locks on the object, or with locks already requested but not given, will be queued for service. The database manager will not suspend the waiting application, unless a deadlock is the cause of the wait, or the timeout period for lock wait is exceeded.

If the timeout period is exceeded, the waiting application will receive a -911 SQLCODE with reason code 68. The unit of work will be automatically rolled back by the database manager.

The length of the timeout period is configurable using the database parameter **locktimeout**. The value is specified in seconds. For transaction environments, the recommended starting value for this parameter is 30 seconds. However, benchmarking and tuning may be necessary to determine the optimum value for a given installation.

If the parameter is set to -1, which is the default, lock timeout is disabled and the waiting application will not receive a timeout.

Although locktimeout values other than -1 will prevent an application from an endless wait, they do not remove the need for long-running applications to commit or roll back statements within reasonable amounts of time. Large applications that require massive amounts of data or require extensive time to complete a unit of work should be scheduled during non-prime hours. If this is not possible, the application should be analyzed to determine if the unit of work can be reduced.

SET CURRENT LOCK TIMEOUT



- Assigns a value to the CURRENT LOCK TIMEOUT special register for the current session

```
SET [CURRENT] LOCK TIMEOUT [=] { WAIT | NOT WAIT
| NULL | WAIT [integer-constant] | host-variable }
```

- CURRENT LOCK TIMEOUT can be set anytime while connected to a database
- The register setting is used immediately for subsequent SQL statements until value is set again
- You can read the current register setting with:

```
VALUES CURRENT LOCK TIMEOUT
```

© Copyright IBM Corporation 2007

Figure 8-19. SET CURRENT LOCK TIMEOUT

CF238.3

Notes:

Prior to DB2 UDB v8.2, the lock timeout *for all users and all sessions* was controlled by one configuration parameter:

Lock timeout (sec)	(LOCKTIMEOUT) = -1
--------------------	--------------------

Now each session is able to control its own lock timeout and change the value as often as needed. If the session does not SET its own lock timeout, the default from the configuration file applies. If it does SET its own lock timeout, the setting applies until a new value is set. The lock timeout applies to *each* lock requested. If the timeout period expires, an error is returned to the program requesting the lock.

The new CURRENT LOCK TIMEOUT special register provides a more granular approach to lock timeout for applications that require varying lock wait times, depending on the business function.

If CURRENT LOCK TIMEOUT is not set, the default database configuration parameter is used instead. When it is set, the value overrides the default database configuration parameter until, and if, SET CURRENT LOCK TIMEOUT = NULL is used.

Functionality

The SET CURRENT LOCK TIMEOUT statement changes the value of the CURRENT LOCK TIMEOUT special register. It is not under transaction control, and is persistent across COMMIT boundaries for the session.

The statement can be embedded in an application program or issued interactively. It is an executable statement that can be dynamically prepared.

The statement applies to the current session and no authorization is needed to execute this statement.

```
SET [CURRENT] LOCK TIMEOUT [=] { WAIT | NOT WAIT | NULL
| WAIT [integer-constant] | host-variable }
```

The specified value must be an integer between -1 and 32767, inclusive, representing the number of seconds maximum that the application is prepared to wait for a lock — or the value NULL to reset the register to the default setting.

WAIT

Specifies a CURRENT LOCK TIMEOUT value of -1, which means that the database manager is to wait until a lock is released, or a deadlock is detected.

NOT WAIT

Specifies a CURRENT LOCK TIMEOUT value of 0, which means that the database manager is not to wait for locks that cannot be obtained, and an error will be returned.

NULL

Specifies that the CURRENT LOCK TIMEOUT value is to be unset, and that the value of the locktimeout database configuration parameter is to be used when waiting for a lock.

The value that is returned for the special register will change as the value of locktimeout changes.

WAIT integer-constant

Specifies an integer value between -1 and 32767. A value of -1 is equivalent to specifying the WAIT keyword without an integer value. A value of 0 is equivalent to specifying the NOT WAIT clause. If the value is between 1 and 32767, the database manager will wait that number of seconds (if a lock cannot be obtained) before an error.

host-variable

A variable of type INTEGER. The value must be between -1 and 32767. If host-variable has an associated indicator variable, and the value of that indicator variable specifies a null value, the CURRENT LOCK TIMEOUT value is unset. This is equivalent to specifying the NULL keyword.

For compatibility with IBM Informix database servers:

- MODE can be specified in place of TIMEOUT
- TO can be specified in place of the equals (=) operator

- SET LOCK MODE WAIT can be specified in place of SET CURRENT LOCK TIMEOUT WAIT
- SET LOCK MODE NO WAIT can be specified in place of SET CURRENT LOCK TIMEOUT NOT WAIT.

An updated value of the special register takes effect immediately upon successful execution of this statement. Because the special register value that is to be used during statement execution is fixed at the beginning of statement execution, an updated value of the CURRENT LOCK TIMEOUT special register will only apply to statements that start execution after the SET statement has completed successfully.

Examples

Example 1: Set the lock timeout value to wait for 30 seconds before returning an error.

```
SET CURRENT LOCK TIMEOUT 30
```

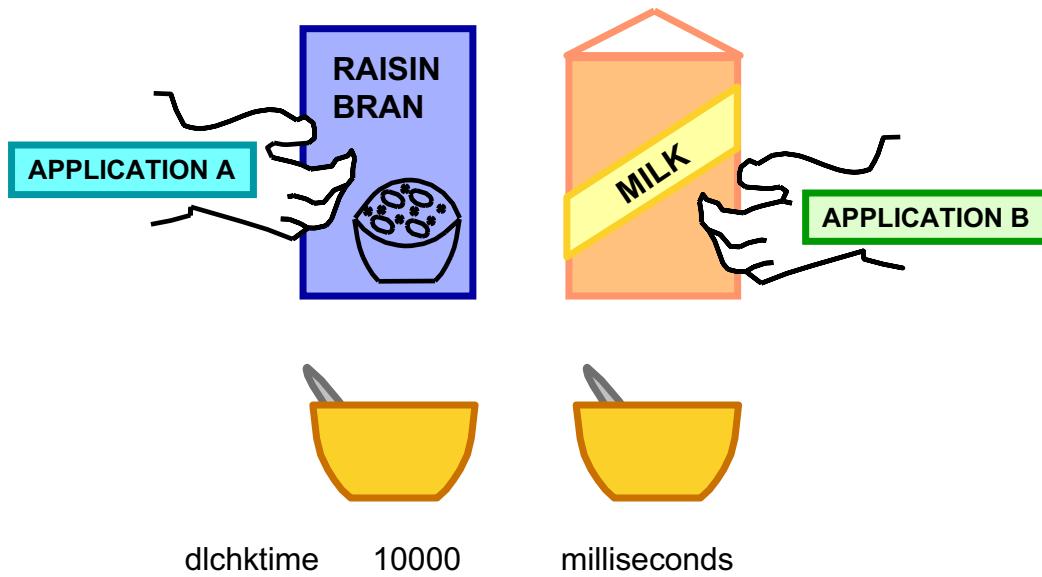
Example 2: Unset the lock timeout value, so that the LOCKTIMEOUT database configuration parameter value will be used instead.

```
SET CURRENT LOCK TIMEOUT NULL
```

Deadlock causes and detection



UNIT OF WORK —
DELETE SOME CEREAL AND MILK



© Copyright IBM Corporation 2007

Figure 8-20. Deadlock causes and detection

CF238.3

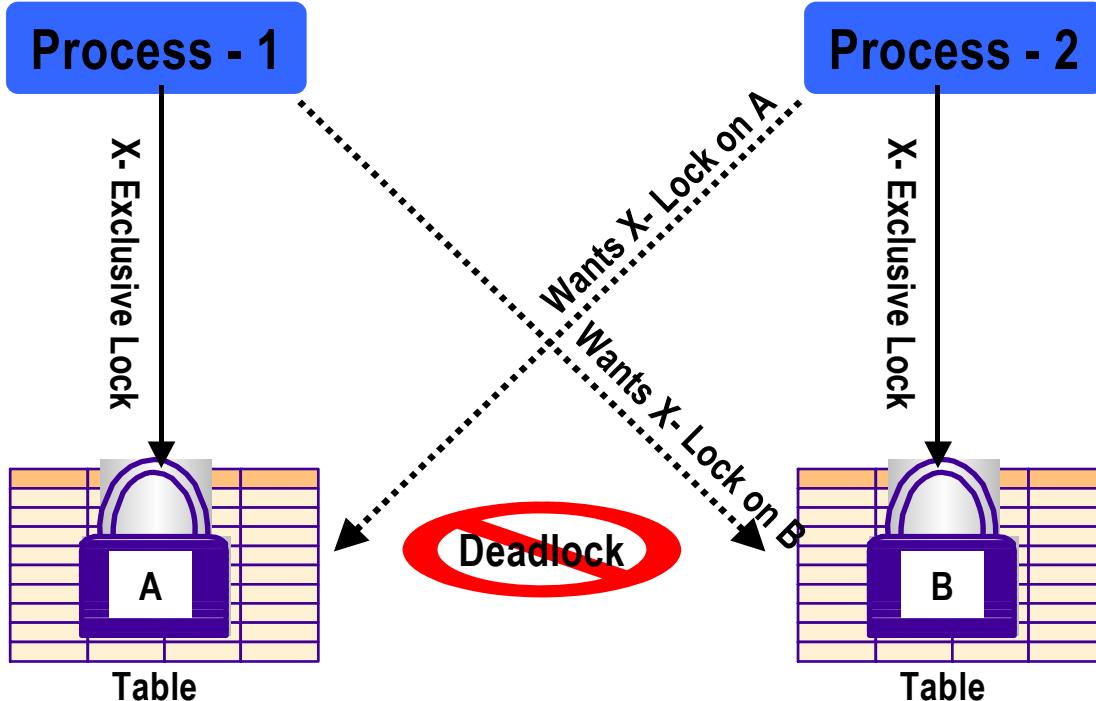
Notes:

A deadlock occurs when applications cannot complete a unit of work due to conflicting lock requirements that cannot be resolved until the unit of work is completed.

The visual illustrates the concept of deadlocks. The unit of work that both application A and application B need to complete before committing is to get a bowl of cereal with milk. For the sake of simplicity, assume there is only enough milk and cereal left for a single bowl. (Another way for the scenario to work is to assume the milk represents a single row and the cereal represents a single row.)

1. Application A obtains an X lock on the cereal.
2. Application B obtains an X lock on the milk.
3. Application A wants an X lock on the milk but cannot obtain it until application B commits.
4. Application B wants an X lock on the cereal but cannot obtain it until application A commits.

Neither application can proceed to a commit point.



This type of deadlock, caused by accessing objects in reverse order, can be reduced by establishing rules of access at an installation for highly used objects. In this particular example, if both applications access the cereal first, followed by the milk, the first application to process would obtain an X lock on the cereal and prevent the other from continuing at that point. The application possessing the X lock on the cereal could proceed to get the X lock on the milk and complete the unit of work.

Other contributors to deadlocks:

- Repeatable Read and Read Stability (major)
- Lock Escalation (major)
- Lock Conversion (minor)
- Catalog Modification (moderate)
- Referential Constraint Enforcement (minor)

The ultimate cause of all deadlocks is poor programming. With proper design, deadlocks are impossible.

Deadlock Detector

Deadlocks are handled by a background process called the *deadlock detector*. If a deadlock is detected, a victim is selected, then the victim is automatically rolled back and returned a negative SQL code (-911) and reason code 2. Rolling back the victim releases locks and should allow other processes to continue.

The deadlock check interval (DLCHKTIME) defines the frequency at which the database manager *checks* for deadlocks among all the applications connected to a database.

- Time_interval_for_checking_deadlock = dlchktime
- Default [Range]: 10,000 (10 seconds) [1000–600,000]
- Unit of measure: milliseconds

dlchktime — Configuration parameter that sets the deadlock check interval. This value is designated in milliseconds and determines the interval for the asynchronous deadlock checker to *wake up* for the database. The valid range of values is 1000 to 600,000 milliseconds. Setting this value high will increase the time that applications will wait before a deadlock is discovered, but the cost of executing the deadlock checker is saved. If the value is set low, deadlocks are detected quickly, but a decrease in run-time performance could be experienced due to checking. The default value corresponds to 10 seconds. This parameter is configurable online.

Checkpoint

Exercise — Unit Checkpoint

- ___ 1. You are evaluating the locking strategy of a program that reads and changes data. The program reads or changes a limited number of rows in the target tables and should run during the time when heavy use of the system is expected by other transactions. What lock modes best support this level of required concurrency at the table level?

- ___ 2. A programmer requests your advice. An application that will run during a batch cycle (no other concurrent requests against the target table) will be written to update a large portion of the table. Explain indicates an IX lock for the table space and table. Would you suggest any technique for the programmer to investigate, and if so, what is the technique?

- ___ 3. An airline requires an application that will determine available seats on a particular flight to support the reservation system. The application should list the seats available that meet a general grouping (for example, aisle seat toward the front of the plane) so that reservations can suggest a seat when booking a passenger. However, the application should not prevent update to the row concerning the seat while the passenger is considering possible options. (In other words, the seat is not held for the passenger while the passenger is thinking; in fact, the cursor is closed after producing the answer set.) Assume that the airline uses conversational transactions (that is, the transaction does not commit prior to displaying a screen). What isolation level would be most suitable?

- ___ 4. Assume that the airline described in the previous question now wants to ensure that any seat being shown to the user will be available, should that user wish to reserve it. What isolation level would be most suitable?

- ___ 5. You are the database administrator for a large financial institution. Most processing requirements of the business are accomplished by using an isolation level of Cursor Stability. However, assume that government regulations dictate that a status report of an account table must be generated at 12:00 p.m. on the last day of business each month. Furthermore, the regulations stipulate that the state of all rows is not permitted to change from the start of the report application until it is completed. How would you suggest meeting this requirement?

6. You receive several phone calls from users of your database complaining about a high number of deadlocks and timeouts. You begin investigating by taking a snapshot with the monitor and discover a high number of lock escalations. You know that you have configured the locklist and maxlocks parameters to handle your normal application requirements. What would be your next area of investigation?
-

Unit summary

Having completed this unit, you should be able to:

- Explain why locking is needed
- List objects that can be locked
- Describe and discuss the various lock modes and their compatibility
- Explain four different levels of data protection
- Set isolation level and lock time out for current activity
- Explain lock conversion and escalation
- Describe the situation that causes deadlocks

© Copyright IBM Corporation 2007

Figure 8-21. Unit summary

CF238.3

Notes:

Unit 9. Problem determination

What this unit is about

This unit provides information about DB2 monitoring and problem determination tools.

What you should be able to do

After completing this unit, you should be able to:

- Collect information for problem analysis and resolution
- Use error logs for basic problem analysis
- Describe four types of monitors:
 - Snapshot Monitor
 - Event Monitor
 - Activity Monitor
 - Health Monitor
- Describe the function of EXPLAIN and use this facility to assist basic analysis
- Use a series of basic commands to better work with connections and sessions
- Retrieve statistics and other information from a running DB2 instance

How you will check your progress

Accountability:

- Checkpoint
- Lab exercises

References

Administration Guide

System Monitor Guide and Reference

Unit objectives



After completing this unit, you should be able to:

- Collect information for problem analysis and resolution
- Use error logs for basic problem analysis
- Describe four types of monitors:
 - Snapshot Monitor
 - Event Monitor
 - Activity Monitor
 - Health Monitor
- Describe the function of EXPLAIN and use this facility to assist basic analysis
- Use a series of basic commands to better work with connections and sessions
- Retrieve statistics and other information from a running DB2 instance

© Copyright IBM Corporation 2007

Figure 9-1. Unit objectives

CF238.3

Notes:

9.1 Problem solving

How to solve a problem



- Understand the nature of the problem
 - Common problem types
- Determine the cause of the conditions
 - Is the problem reproducible?
 - Was it a one time occurrence?
- Keep detailed notes — your problem description should include:
 - ALL error codes/error conditions
 - ALWAYS include the reason code, if applicable
 - The actions which preceded the error
 - A reproducible scenario, if possible

© Copyright IBM Corporation 2007

Figure 9-2. How to solve a problem

CF238.3

Notes:

In order to solve the problem, you must understand the nature of the problem, and determine the cause of the conditions.

Common problem types are:

- Unexpected messages/SQL codes
- Abends (*abnormal endings*) of application or DBM
- Loops and hangs
- Traps
- Database/data corruption
- Data loss
- Incorrect or inconsistent documents
- Incorrect output
- Install failure
- Performance
- Usability

Important points to consider are:

- Is the problem reproducible? If so, how is it reproducible (by the clock, a certain application)?
- Was it a one time occurrence? What were the operating conditions at the time?

A comprehensive problem description should include:

- ALL error codes/error conditions
- ALWAYS include the reason code, if applicable
- The actions which preceded the error
- A reproducible scenario, if possible
- System and application log files, if available

Diagnostic data and data checklist

- What information is generally required to analyze a DB2 problem and determine the solution?
 - ✓ SQLCODE / SQLSTATE / reason codes / system error codes
 - ✓ A good problem description
 - ✓ System-time of the error (to pin-point the time in logs)
 - ✓ Include the SQL code and any associated reason code
 - ✓ A description of the actions preceding the error
 - ✓ GET DBM CFG / GET DB CFG listings
 - ✓ **db2diag.log** entries (see next visuals)
 - ✓ Dump files listed in the db2diag.log / trap files in the DIAGPATH
 - ✓ Operating system software level, and service patches
 - ✓ Hardware model and peripheral equipment
 - ✓ Database (DB2) version and fixpak level
 - ✓ Other available data

© Copyright IBM Corporation 2007

Figure 9-3. Diagnostic data and data checklist

CF238.3

Notes:

To troubleshoot the problem correctly, you need to collect the information required to analyze a DB2 problem and determine the solution.

Use the **db2diag.log** file to view captured diagnostic information.

- If reproducible, setting the DIAGLEVEL to 4 and recapturing the information is recommended.
- Any dump files mentioned in **db2diag.log (pid/tid.node)**.
- Any traceback/trap files in the DIAGPATH (**tpid/tid.node** or ***.trp**). You must check for them manually—know the format!
- With workgroup edition (WE) or enterprise server edition (ESE), send all files in the DB2DUMP directory.
 - To reduce the number of viewed files, clean up this directory on a regular basis.
 - Also copy and truncate the **db2diag.log** file.

First Failure Data Capture (FFDC)



DBM configuration parameters

NOTIFY LEVEL — (0-4, default 3)

- 0 — No administrative notification messages captured
- 1 — Fatal or unrecoverable errors
- 2 — Immediate action required
- 3 — Important information, no immediate action required
- 4 — Informational messages

DIAGLEVEL — (0-4, default 3)

- 0 — No diagnostic data captured
- 1 — Severe errors only
- 2 — All errors
- 3 — All errors and warnings
- 4 — All errors, warnings, and informational messages



DIAGPATH — valid directory path, may contain

- ✓ Notification file
- ✓ Error log
- ✓ Dump files
- ✓ Trap files

© Copyright IBM Corporation 2007

Figure 9-4. First Failure Data Capture (FFDC)

CF238.3

Notes:

When significant events occur, DB2 writes information to the administration notification log. The information is intended for use by database and system administrators. Many notification messages provide additional information to supplement the SQLCODE that is provided. The type of event and the level of detail of the information gathered are determined by the NOTIFYLEVEL configuration parameter. However, detailed diagnostic information is not written to this log.

NOTIFYLEVEL specifies the type of administration notification messages that are written to the administration notification log. On UNIX platforms, the administration notification log is a text file called *instance.nfy*. On Windows, all administration notification messages are written to the Event Log. The errors can be written by DB2, the Health Monitor, the Capture and Apply programs, and user applications.

Valid values for this parameter are:

0 — No administration notification messages captured. (This setting is not recommended.)

- 1 — Fatal or unrecoverable errors. Only fatal and unrecoverable errors are logged. To recover from some of these conditions, you may need assistance from DB2 service.
- 2 — Immediate action required. Conditions are logged that require immediate attention from the system administrator or the database administrator. If the condition is not resolved, it could lead to a fatal error. Notification of very significant, non-error activities (for example, recovery) may also be logged at this level. This level will capture Health Monitor alarms.
- 3 — Important information, no immediate action required. Conditions are logged that are non-threatening and do not require immediate action but may indicate a non-optimal system. This level will capture Health Monitor alarms, Health Monitor warnings, and Health Monitor attentions.
- 4 — Informational messages.

The administration notification log includes messages having values up to and including the value of *notifylevel*. For example, setting *notifylevel* to 3 will cause the administration notification log to include messages applicable to levels 1, 2, and 3.

For a user application to be able to write to the notification file or Windows Event Log, it must call the db2AdminMsgWrite API.

Recommendation: You may wish to increase the value of this parameter to gather additional problem determination data to help resolve a problem. Note that you must set *notifylevel* to a value of 2 or higher for the Health Monitor to send any notifications to the contacts defined in its configuration.

Error log

Diagnostic information about errors is recorded in the db2diag.log text file. This information is used for problem determination and is intended for DB2 customer support.

The type of diagnostic errors recorded in the db2diag.log is determined by **DIAGLEVEL**. Valid values are:

- 0 — No diagnostic data captured
- 1 — Severe errors only
- 2 — All errors
- 3 — All errors and warnings
- 4 — All errors, warnings, and informational messages

It is the *diagpath* configuration parameter that is used to specify the directory that will contain the error file, event log file (on Windows NT only), alert log file, and any dump files that may be generated based on the value of the *diaglevel* parameter.

Diagnostic file location

DIAGPATH allows you to specify the fully qualified path for DB2 diagnostic information. This directory could possibly contain dump files, trap files, an error log, a notification file, and an alert log file, depending on your platform.

If this parameter is null, the diagnostic information will be written to files in one of the following directories or folders:

- For supported Windows environments:
 - If the DB2INSTPROF environment variable or keyword is **not** set, information will be written to x:\IBM\SQLLIB \%DB2INSTANCE%, where x:\IBM\SQLLIB is the drive reference and directory specified in the DB2PATH registry variable or environment variable, and %DB2INSTANCE% is the name of the instance owner.
- Note:** The directory does not have to be named SQLLIB. By default, DB2 UDB installs on Windows into \Program Files\IBM\SQLLIB.
- If the %DB2INSTPROF% environment variable or keyword is set, information will be written to x:\%DB2INSTPROF% \%DB2INSTANCE%, where DB2INSTPROF is the name of the instance profile directory and %DB2INSTANCE% is the name of the instance owner.
- For UNIX-based environments: \$INSTHOME/sqllib/db2dump, where \$INSTHOME is the home directory of the instance owner.



Note

You should clean out the dump directory periodically to keep it from becoming too large.

DB2 Administration Server First Failure Data Capture

The DB2 Administration Server (DAS) also collects information for first failure data capture. First failure data capture (FFDC) is a general term applied to the set of diagnostic information the DB2 administration server captures automatically when errors occur. This information reduces the need to reproduce errors to get diagnostic information. The diagnostic information is contained in a single location.

The information captured by the DB2 administration server FFDC includes:

- Administration notification logs — When an event occurs, the DB2 administration server writes information to the DB2 administration server log file, db2dasdiag.log.
- Dump files — For some error conditions, extra information is logged in external binary dump files named after the failing process ID. These files are intended for use by DB2 Customer Support.
- Trap files — The DB2 administration server generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception. Trap files contain a function flow of the last steps that were run before a problem occurred.

By default, the DB2 administration server FFDC information is placed in the following locations:

- On Windows systems:

- If the DB2INSTPROF environment variable is not set:

db2path\DB2DAS00\dump

where db2path is the path referenced in the DB2PATH environment variable, and DB2DAS00 is the name of the DAS service. The DAS name can be obtained by typing the **db2admin** command without any arguments.

- If the DB2INSTPROF environment variable is set:

x:\%DB2INSTPROF%\DB2DAS00\dump

where x: is the drive referenced in the %DB2PATH% environment variable, db2instprof is the instance profile directory, and DB2DAS00 is the name of the DAS service.

- On UNIX-based systems:

\$DASHOME/das/dump

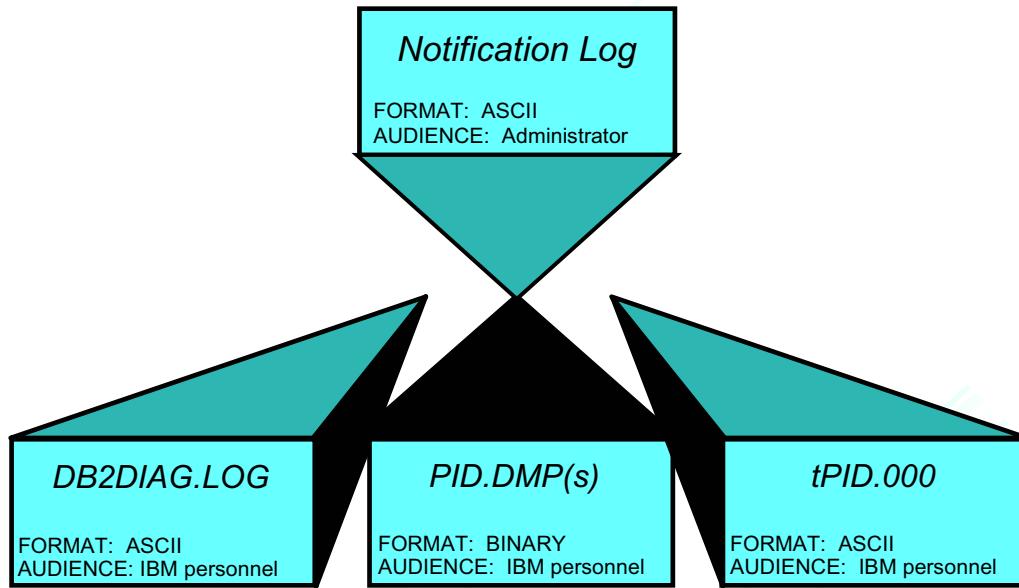
where \$DASHOME is the home directory of the DAS user.



Note

You should clean out the dump directory periodically to keep it from becoming too large.

FFDC structure



© Copyright IBM Corporation 2007

Figure 9-5. FFDC structure

CF238.3

Notes:

First-failure data capture (FFDC) is a general term applied to the set of diagnostic information that DB2 captures automatically when errors occur. This information reduces the need to reproduce errors to get diagnostic information. The DIAGPATH variable, specified in the database manager configuration, gives the fully qualified path to the FFDC storage directory. The DIAGLEVEL and NOTIFYLEVEL configuration parameters control the detail of information you receive in the logs.

The information captured by FFDC includes the following:

- Administration Notification Logs — When significant events occur, DB2 writes information to the administration notification log. The information is intended for use by database and system administrators. Many notification messages provide additional information to supplement the SQLCODE that is provided. The type of event and the level of detail of the information gathered are determined by the NOTIFYLEVEL configuration parameter.
- db2diag.log — Diagnostic information about errors is recorded in this text log file. This information is used for problem determination and is intended for DB2 customer

support. The level of detail of the information is determined by the DIAGLEVEL configuration parameter.

- Dump files — For some error conditions, extra information is logged in external binary files named after the failing process ID. These files are intended for use by DB2 customer support.
- Trap files — The database manager generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception.
- Core files (UNIX only) — When DB2 terminates abnormally, the operating system generates a core file. The core file is a binary file that contains information similar to the DB2 trap files. Core files may also contain the entire memory image of the terminated process.

Interpreting the Notification Log

TIME OF THE MESSAGE

```

2006-09-23-12.53.00.529019  Instance:inst01  Node:000
PID:901370(db2star2)  TID:1  Appid:none
base sys utilities  startdbm Probe:911

```

COMPONENT Database manager FUNCTION NAME ed.

INSTANCE NAME

```

2006-09-23-12.59.54.18814  Instance:inst01  Node:000
PID:909554(db2agent (SAMPLE))  TID:1  Appid:*LOCAL.inst01.0C0233165917
buffer pool services  sqlbCreateBufferPool Probe:130  Database:SAMPLE

```

ADM6053W The CREATE BUFFERPOOL statement for buffer pool "BP4K" (ID "2") could not be performed immediately because not enough free memory existed in the database shared memory. The buffer pool will be created on the next database restart. Refer to the documentation for SQLCODE 20189.

Explains the message

```

2006-09-23-13.57.31.897606  Instance:inst01  Node:000
PID:655610(db2stop2)  TID:1  Appid:none
base sys utilities  stopdbm Probe:911

```

```

ADM7514W Database manager has stopped.

```

© Copyright IBM Corporation 2007

Figure 9-6. Interpreting the Notification Log

CF238.3

Notes:

The db2diag.log and administration notification log are files that contain text information logged by DB2. They are located in the directory specified by the DIAGPATH database manager configuration parameter. On Windows NT, Windows 2000, and Windows XP systems, the DB2 administration notification log is found in the event log and can be reviewed through the Windows Event Viewer.

The information that DB2 records in the administration logs is determined by the settings for DIAGLEVEL and NOTIFYLEVEL.

Use a text editor to view the file on the machine where you suspect a problem to have occurred. The most recent events recorded are the furthest down the file. Generally, each entry contains the following parts:

- A timestamp
- The location reporting the error. Application identifiers allow you to match up entries pertaining to an application on the logs of servers and clients.

A diagnostic message (usually beginning with DIA or ADM) explaining the error.

- Any available supporting data, such as SQLCA data structures and pointers to the location of any extra dump or trap files.

If the database is behaving normally, this type of information is not important and can be ignored.

- The Administration logs *grow continuously*. When they get too large, back them up and then erase the file. A new set of files is generated automatically the next time they are required by the system.

The figure shows the start of an instance, and a message generated because a buffer pool was created for which there was not enough free memory. Then the notification log indicates that the instance was stopped.

Other types of messages that will be placed in the administration notification log:

- Lock timeout information. This will be captured at notification level 4, including locked object, lock mode, and application holding the lock. Deadlock information will also be available if notification level 4 is set.
- Lock escalations
- Abnormal termination to Governor
- Full event monitor log file and the event monitor turning off
- Current set of active units of work exceed primary log files when infinite logging is turned on
- Disk or file system for active log path being full
- Primary or secondary log path being unavailable
- Beginning of database restart operation

Some DIAGLEVEL 4 considerations

- DIAGLEVEL 4 logs more information than lower levels
- This causes DB2 to run a little slower, but only during the following times:
 - During an error condition
 - During **db2start** processing
 - During an initial connect to a database
- Balance out the extra data provided with the decreased response time to determine the best setting for the environment



Warning

Be careful when using DIAGLEVEL 4. Do not set it at that level during *normal* daily activities; use it only as a troubleshooting aid. Remember, **db2diag.log** grows in size as it is used, so it is a good idea to copy it to a safe location and truncate the original file periodically.

Diagnostic Log Analysis Tool: ***db2diag***



- Command line tool for filtering and formatting ***db2diag.log*** files:
 - Complex set of options, reflecting wide variety of filtering and analysis
 - Ability to use *grep*-like filtering to reduce the number of records returned
- Profile registry variables (**db2set**) and DB or DBM configuration parameters are now logged

Note:

The UNIX “grep” command allows records to pass from input to output if they match a provided “pattern”

The equivalent DOS/Windows command is “find”

© Copyright IBM Corporation 2007

Figure 9-7. Diagnostic Log Analysis Tool: ***db2diag***

CF238.3

Notes:

The ***db2diag.log*** format has been improved in a number of ways with DB2 UDB 8.2. This log file structure is now easier to read manually and easier to parse in software. The improvements include:

- Each entry has a complete set of header information
- Message and data portions of the logging record are clearly marked, making the logging record easier to use and to understand
- Timestamps (with time zone) are included in each record
- Each field has a clear field name (in uppercase)
- Header and message field line lengths are restricted to 80 characters
- New fields have been added, most notably a severity-level field to help you find the most important entries

Other changes have been made as well, such as changing the database field name to ***DB***.

Analysis Tool (db2diag)

db2diag is a new command line tool for filtering and formatting **db2diag.log** files. You can use this tool to filter diagnostic log files that use the new message format for V8.2.

This tool has a wide range of options that you can use to modify the output to suit your needs. Among other options, you can indicate which fields to display, use a “grep”-like filter to reduce the number of records, and have the empty fields omitted.

Example output from db2diag

Lines of particular interest are shown in **bold**. The actual file naturally does not have highlighted text. This information was gathered by executing the following steps in sequence:

- Removing the old **db2diag.log** file (this can be done at any time, but would be generally done by renaming and archiving the file) — remember that you should do this as a normal administrative operational step from time to time as the diagnostic log file grows rapidly on a production system
- Stopping database manager (**db2stop**) and then starting the current database manager instance (**db2start**)
- Altering the DBM configuration file for the instance (DB2) by changing the DIAGLEVEL from “3” to “4” (db2 update dbm cfg using diaglevel 4) — to illustrate that these changes are now recorded in the diagnostic log:

```
2006-10-30-18.23.57.726000-300 I1H929          LEVEL: Event
PID      : 3648           TID  : 3884           PROC  : DB2STOP.EXE
INSTANCE: DB2             NODE : 000
FUNCTION: DB2 UDB, RAS/PD component, pdLogInternal, probe:120
START    : New Diagnostic Log file
DATA #1 : Build Level, 128 bytes
Instance "DB2" uses "32" bits and DB2 code release "SQL09010"
with level identifier "02010107".
Informational tokens are "DB2 v9.1.0.356", "s060629", "NT32", Fix Pack "0".
DATA #2 : System Info, 1564 bytes
System: WIN32_NT IBM-GLENMULES Service Pack 2 5.1 x86 Family 6, model 9, stepping 5
CPU: total:1 online:1 Cores per socket:1 Threading degree per core:1
Physical Memory(MB) : total:2047 free:134 available:134
Virtual   Memory(MB) : total:3942 free:1788
Swap      Memory(MB) : total:1895 free:1654
Information in this record is only valid at the time when this file was
created (see this record's time stamp)

2006-10-30-18.23.57.726000-300 I933H1527          LEVEL: Event
PID      : 3648           TID  : 3884           PROC  : DB2STOP.EXE
INSTANCE: DB2             NODE : 000
FUNCTION: DB2 UDB, base sys utilities, sqleStartStopSingleNode, probe:1130
DATA #1 : String, 30 bytes
C:\IBM\SQLLIB\bin\DB2STOP2.EXE
DATA #2 : Hexdump, 256 bytes
0x0012AD24 : 433A 5C49 424D 5C53 514C 4C49 425C 6269      C:\IBM\SQLLIB\bi
0x0012AD34 : 6E5C 4442 3253 544F 5032 2E45 5845 004E      n\DB2STOP2.EXE.N
0x0012AD44 : 4F4D 5347 0053 4E00 0000 0000 0000 0000      OMSG.SN.....
```

```
0x0012AD54 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012AD64 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012AD74 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012AD84 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012AD94 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012ADA4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012ADB4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012ADC4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012ADD4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012ADE4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012ADF4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012AE04 : 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012AE14 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

2006-10-30-18.23.58.086000-300 I2462H390 LEVEL: Event
PID : 3496 TID : 3808 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:210
DATA #1 : String, 29 bytes
Deactivate phase is bypassed.
DATA #2 : Hexdump, 4 bytes
0x04BCFEB : 0100 0000

2006-10-30-18.23.58.086000-300 I2854H410 LEVEL: Event
PID : 3496 TID : 3808 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:230
DATA #1 : String, 49 bytes
Deactivate phase is completed, preparing to stop.
DATA #2 : Hexdump, 4 bytes
0x04BCFEB : 0100 0000

2006-10-30-18.23.58.086000-300 I3266H297 LEVEL: Event
PID : 3496 TID : 3808 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:240
DATA #1 : String, 26 bytes
Stop phase is in progress.

2006-10-30-18.23.58.086000-300 I3565H312 LEVEL: Event
PID : 3496 TID : 3808 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:250
DATA #1 : String, 41 bytes
Requesting system controller termination.

2006-10-30-18.23.58.106000-300 I3879H405 LEVEL: Warning
PID : 3496 TID : 4112 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, routine_infrastructure, sqlerKillAllFmps, probe:5
MESSAGE : Bringing down all db2fmp processes as part of db2stop
DATA #1 : Hexdump, 4 bytes
0x03EEFA64 : 0000 0000

2006-10-30-18.23.58.116000-300 I4286H314 LEVEL: Event
PID : 3496 TID : 3808 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:260
DATA #1 : String, 43 bytes
System controller termination is completed.

```

2006-10-30-18.23.58.136000-300 I4602H385           LEVEL: Event
PID      : 3496                      TID  : 3808          PROC : db2syscs.exe
INSTANCE: DB2                       NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:280
DATA #1 : String, 24 bytes
There is no active EDUs.
DATA #2 : Hexdump, 4 bytes
0x04BCFE64 : 0000 0000 .....  

2006-10-30-18.23.58.256000-300 E4989H311           LEVEL: Event
PID      : 3496                      TID  : 3808          PROC : db2syscs.exe
INSTANCE: DB2                       NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:911
MESSAGE : ADM7514W Database manager has stopped.
STOP     : DB2 DBM  

2006-10-30-18.24.06.238000-300 I5302H1527           LEVEL: Event
PID      : 4824                      TID  : 3604          PROC : DB2STOP.EXE
INSTANCE: DB2                       NODE : 000
FUNCTION: DB2 UDB, base sys utilities, sqleStartStopSingleNode, probe:1130
DATA #1 : String, 30 bytes
C:\IBM\SQLLIB\bin\DB2STOP2.EXE
DATA #2 : Hexdump, 256 bytes
0x0012AD24 : 433A 5C49 424D 5C53 514C 4C49 425C 6269 C:\IBM\SQLLIB\bi
0x0012AD34 : 6E5C 4442 3253 544F 5032 2E45 5845 004E n\DB2STOP2.EXE.N
0x0012AD44 : 4F4D 5347 0053 4E00 0000 0000 0000 0000 OMSG.SN.....
0x0012AD54 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AD64 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AD74 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AD84 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AD94 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012ADA4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012ADB4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012ADC4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012ADD4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012ADE4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012ADF4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE04 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE14 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
  

2006-10-30-18.24.14.460000-300 I6831H1528           LEVEL: Event
PID      : 5788                      TID  : 5540          PROC : DB2START.EXE
INSTANCE: DB2                       NODE : 000
FUNCTION: DB2 UDB, base sys utilities, sqleStartStopSingleNode, probe:1130
DATA #1 : String, 30 bytes
C:\IBM\SQLLIB\bin\DB2STAR2.EXE
DATA #2 : Hexdump, 256 bytes
0x0012ADB4 : 433A 5C49 424D 5C53 514C 4C49 425C 6269 C:\IBM\SQLLIB\bi
0x0012ADC4 : 6E5C 4442 3253 5441 5232 2E45 5845 004E n\DB2STAR2.EXE.N
0x0012ADD4 : 4F4D 5347 0000 0000 0000 0000 0000 0000 OMSG.....
0x0012ADE4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012ADF4 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE04 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE14 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE24 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE34 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE44 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE54 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE64 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE74 : 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x0012AE84 : 0000 0000 0000 0000 0000 0000 0000 0000 .....

```

```
0x0012AE94 : 0000 0000 0000 0000 0000 0000 0000 .....  
0x0012AFA4 : 0000 0000 0000 0000 0000 0000 0000 .....
```

2006-10-30-18.15.59.667000-240 E2565H853 LEVEL: Warning
PID : 1744 TID : 2944 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, oper system services, licRequestAccess, probe:1
MESSAGE : ADM12007E There are "63" day(s) left in the evaluation period for
the product "DB2 Enterprise Server Edition". For evaluation license
terms and conditions, refer to the IBM License Agreement in the
try_LA file, located in the following directory:
"C:\IBM\SQLLIB\license\en". If you have licensed this product, please
ensure the license key is properly installed. You can install the
license via using License Center or db2licm command line utility. The
license file can be obtained from your licensed product CD.

2006-10-30-18.24.16.573000-300 E8361H853 LEVEL: Event
PID : 5264 TID : 5192 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
FUNCTION: DB2 UDB, base sys utilities, DB2StartMain, probe:911
MESSAGE : ADM7513W Database manager has started.
START : DB2 DBM
DATA #1 : Build Level, 128 bytes
Instance "DB2" uses "32" bits and DB2 code release "SQL09010"
with level identifier "02010107".
Informational tokens are "DB2 v9.1.0.356", "s060629", "NT32", Fix Pack "0".
DATA #2 : System Info, 1564 bytes
System: WIN32_NT IBM-GLENMULES Service Pack 2 5.1 x86 Family 6, model 9, stepping 5
CPU: total:1 online:1 Cores per socket:1 Threading degree per core:1
Physical Memory(MB): total:2047 free:168 available:168
Virtual Memory(MB): total:3942 free:1929
Swap Memory(MB): total:1895 free:1761

2006-10-30-18.24.41.689000-300 I9216H361 LEVEL: Event
PID : 5264 TID : 5308 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
APPHDL : 0-7 APPID: *LOCAL.DB2.061030232441
FUNCTION: DB2 UDB, config/install, sqlfLogUpdateCfgParam, probe:30
CHANGE : CFG DBM: "Diaglevel" From: "3" To: "4"

2006-10-30-18.24.41.689000-300 I9579H500 LEVEL: Info
PID : 5264 TID : 5308 PROC : db2syscs.exe
INSTANCE: DB2 NODE : 000
APPHDL : 0-7 APPID: *LOCAL.DB2.061030232441
FUNCTION: DB2 UDB, config/install, sqf_upd_krcb_dbm_cfg, probe:200
MESSAGE : Not all changes were successful. Now trying to re-apply the
unsuccessful ones
DATA #1 : Hexdump, 4 bytes
0x04C25808 : 0000 0000

db2diag invoked with the default set of filtering/formatting options
on file "C:\IBM\SQLLIB\DB2\db2diag.log".
Enter "db2diag -h" to get a short description of all available options.

Command Options

<i>filename</i>	one or more space-separated path names of diagnostic logs
-help, -h, ?	help information. To get help on help, try " db2diag -h h "
-filter, -g	case-sensitive search for a list of field-pattern pairs
-gi	case-insensitive search for a list of field-pattern pairs
-gv	case-sensitive invert matching
-gvi, -giv	case-insensitive invert matching
-invert, -v	invert the sense of matching for all filtering options
-exist	record field must exist in order to be processed
-pid	find all records for a list of process IDs
-tid	find all records for a list of thread IDs
-node, -n	find all records for a list of nodes
-error, -e	find all records for a list of errors
-level, -l	find all records for a list of severity levels
-history, -H	display the history of logged records for a time interval
-time, -t	display all the records within a particular time interval
-count, -c	display a count of matching records
-verbose, -V	display all record fields whether they contain data or not
-strict	display records using one "field: value" pair per line
-cbe	display records in the Common Base Event (CBE) format
-fmt	format tool's output using a format string
-output, -o	save output into a file
-follow, -f	continuously display appended records as the file grows
-archive, -A	archive a diagnostic log file
-rc	display descriptions of DB2 error return codes, ZRC or ECF
-ecfid	display function info extracted from the numeric ECF ID

The various options and the option parameters are too detailed to cover here. For further information, consult the *Command Reference* manual.

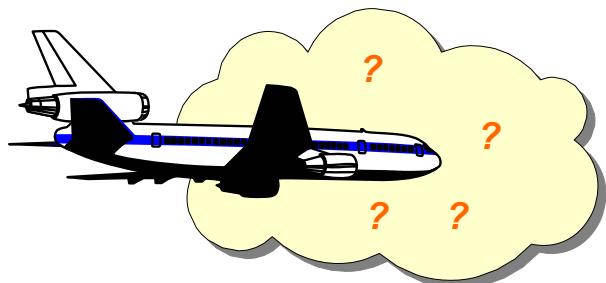
Help Options

- **db2diag -help** — provides a short description of the options
- **db2diag -h option1[,...]** — provides help on one or more options
- **db2diag -h brief** — provides descriptions for all options without examples
- **db2diag -h notes** — provides usage notes and restrictions
- **db2diag -h examples** — provides a small set of examples to get started
- **db2diag -h tutorial** — provides examples for all available options
- **db2diag -h all** — provides the most complete list of options

9.2 Database monitoring

Why use database monitors?

**No need to fly
in or into the dark!!**



DB2 system monitor usage:

- ✓ Improve database and application performance
- ✓ Tune configuration parameters
- ✓ Problem determination
- ✓ Better understand user and application activity

© Copyright IBM Corporation 2007

Figure 9-8. Why use database monitors?

CF238.3

Notes:

To tune a database server effectively, a database administrator needs large amounts of information about the database and the applications accessing the database. The database administrator requires a tool like a database monitor to obtain the required information to improve the database environment (for example, tune configuration parameters).

A database monitoring plan should be developed to help the database administrator decide on the critical success factors of the database application environment. For example, if response time is the most crucial aspect of the database application, then the administrator needs to monitor the number of lock escalations and effective buffer pool usage. Another example could involve monitoring any changes to a particular table.

Keep in mind that there is overhead involved when using database monitors.

Ways to monitor the database

- Snapshot Monitor
 - Query operational database status for an INSTANT in time
- Event Monitor
 - Query operational status OVER time for a specific activity
- Activity Monitor
 - GUI for improving efficiency
- Health Monitor
 - Uses health indicators to gauge the healthfulness of the database manager or database performance
 - Can be accessed from Health Center, Web Health Center, CLP, APIs

© Copyright IBM Corporation 2007

Figure 9-9. Ways to monitor the database

CF238.3

Notes:

With the increasing complexity of DB2 database systems, there exists a strong need for database and database server information to debug problems and monitor performance.

Snapshot monitoring is useful to determine the current state of a database and its applications.

The Activity Monitor tool assists DBAs in improving the efficiency of database performance monitoring, problem determination, and resolution. It helps to diagnose application locking situations and to tune queries for optimal utilization of database resources. It provides easy access to relevant and well-organized monitor data through a set of predefined reports, such as on top CPU-time consuming applications, and on SQL statements with the largest total sort time. For each predefined report, appropriate actions may be recommended to help resolve resource utilization problems, to optimize performance, or to invoke another tool for further investigation. Lock monitor data is also supplied to illustrate the details of lock waiting situations. Application lock chains can be displayed to show lock waiting dependencies. The Activity Monitor is accessible through a GUI interface, CLP, and in the form of stored procedures and UDFs.

Event monitoring is used to gather information about state changes regarding many different event types: databases, buffer pools, connections, tables, statements, transactions, deadlocks, and table spaces.

The amount of memory that can be used for database monitoring (both Snapshot and Event type monitors) is configurable in the DBM configuration file using the *mon_heap_sz* parameter. The memory is allocated at DB2START and the memory is used when monitoring is taking place. If this parameter is set to zero, no monitoring can take place. If this resource is exceeded, the application may receive an SQLCODE, or an error will be logged to the administration notification log, or both.

Monitoring can be invoked through APIs:

- db2MonitorSwitches — turn on or off switches for monitor groups
- db2GetSnapshot — collects monitor data into a user buffer
- db2GetSnapshotSize — estimates buffer size needed
- db2ResetMonitor — resets monitor data to zero

or through CLP:

- GET MONITOR SWITCHES
- GET SNAPSHOT FOR
- RESET MONITOR

The Health Monitor and the Health Center tools provide a *management by exception* capability to DB2 Universal Database by alerting you to potential system health issues. This enables you to address health issues before they become real problems that affect your system's performance.

Snapshot Monitor — who and how?

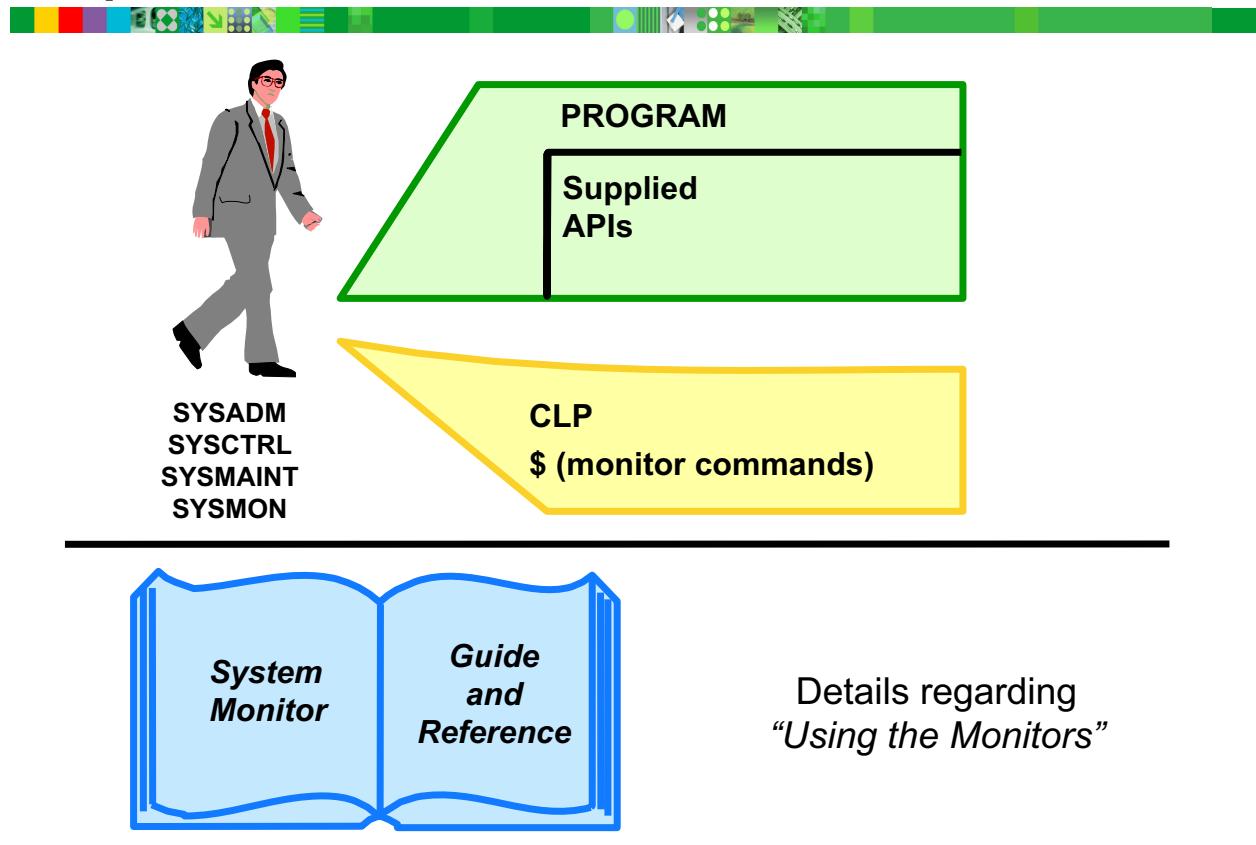


Figure 9-10. Snapshot Monitor — who and how?

CF238.3

Notes:

SYSADM, SYSCTRL, SYSMAINT, or SYSMON authorization is required to use the Snapshot Monitor.

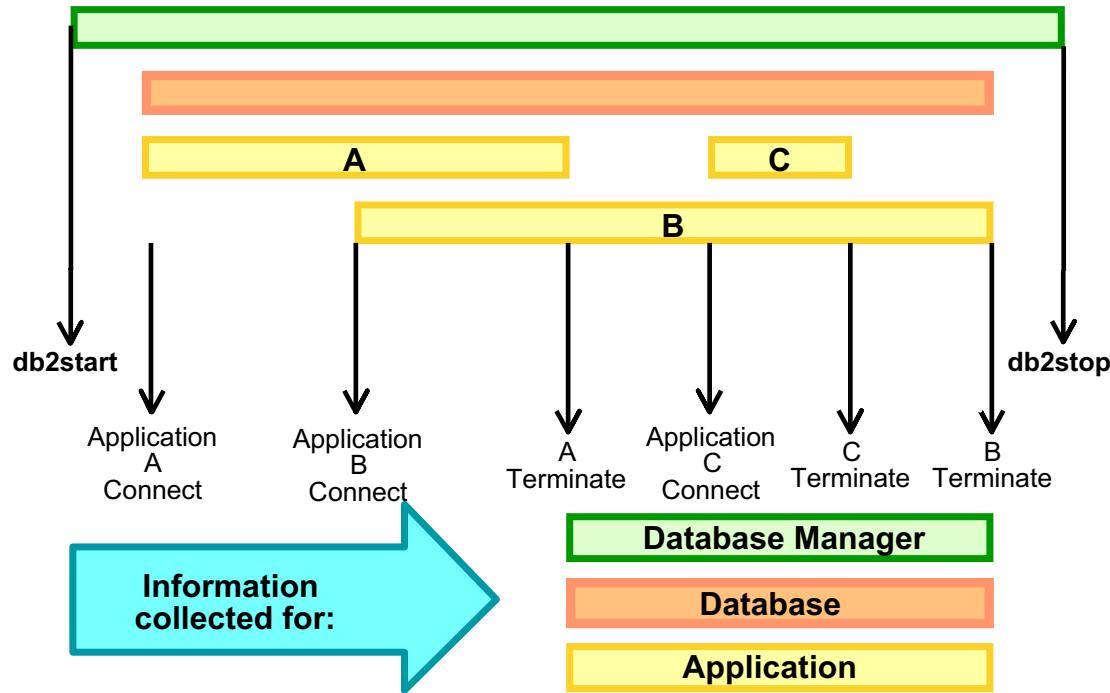
An individual possessing the proper authority can use supplied APIs within an application or issue commands using the Command Line Processor.

The Monitor function of DB2 provides the database administrator with a tool that can assist with analysis of many areas.

In this unit, the basic use of the Snapshot Monitor will be discussed. The System Monitor has several other components and provides a great deal of function.

Details regarding the use of the Monitor are provided in the product documentation noted, as well as education related to performance.

Snapshot Monitor — when?



© Copyright IBM Corporation 2007

Figure 9-11. Snapshot Monitor — when?

CF238.3

Notes:

The Monitor will collect information at different levels within the system.

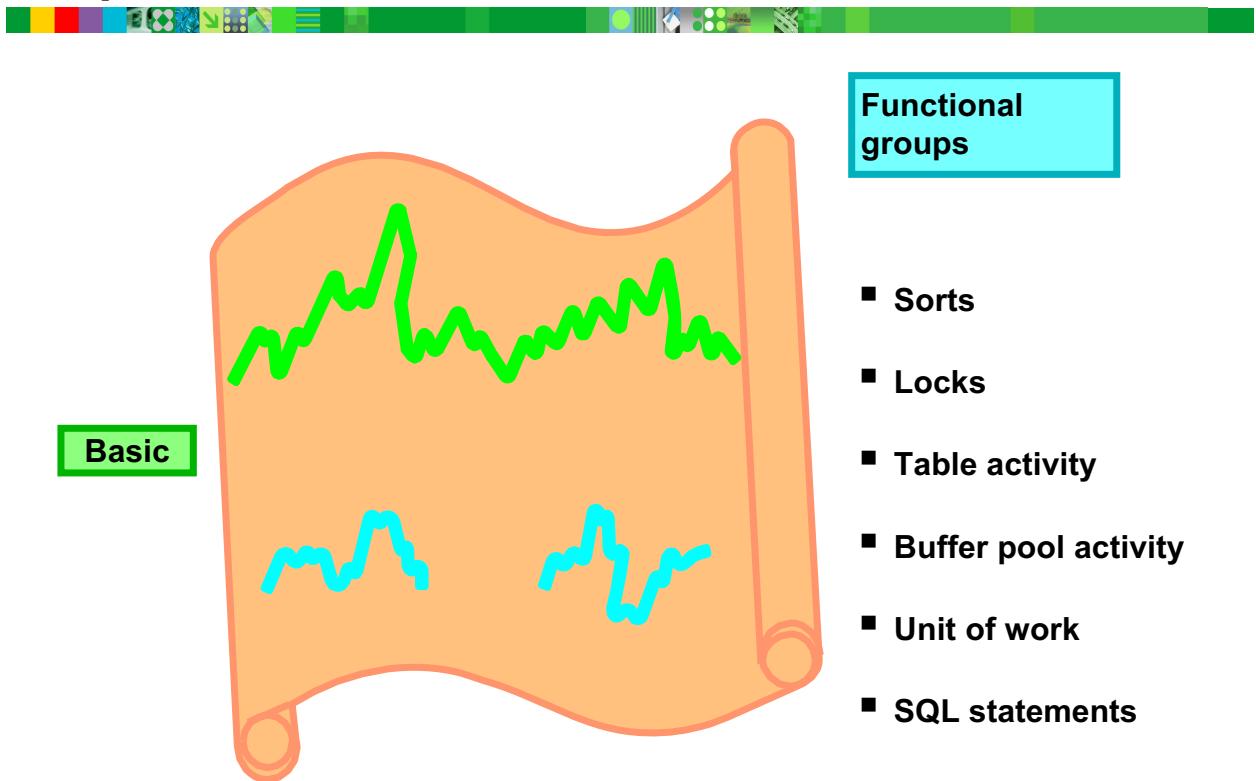
Database manager instance level statistics are collected from the time the instance is started until the instance is stopped.

Database level statistics are collected from the time the first application connects to the database until the last connection is terminated. The visual depicts this monitoring period as a single, continuous process. However, application connections may not overlap as shown in the visual. It may be necessary to invoke an application that establishes a connection and does not disconnect from the database until the monitoring period is over.

Application level statistics are collected from the time the application connects to the database until the application disconnects from the database.

The level of information used depends on the type of analysis to be completed. You should start analysis at the database manager level and proceed to lower levels if necessary. This method enables the database administrator to resolve system tuning problems first, which may in turn solve database and application problems.

Snapshot Monitor — what?



© Copyright IBM Corporation 2007

Figure 9-12. Snapshot Monitor — what?

CF238.3

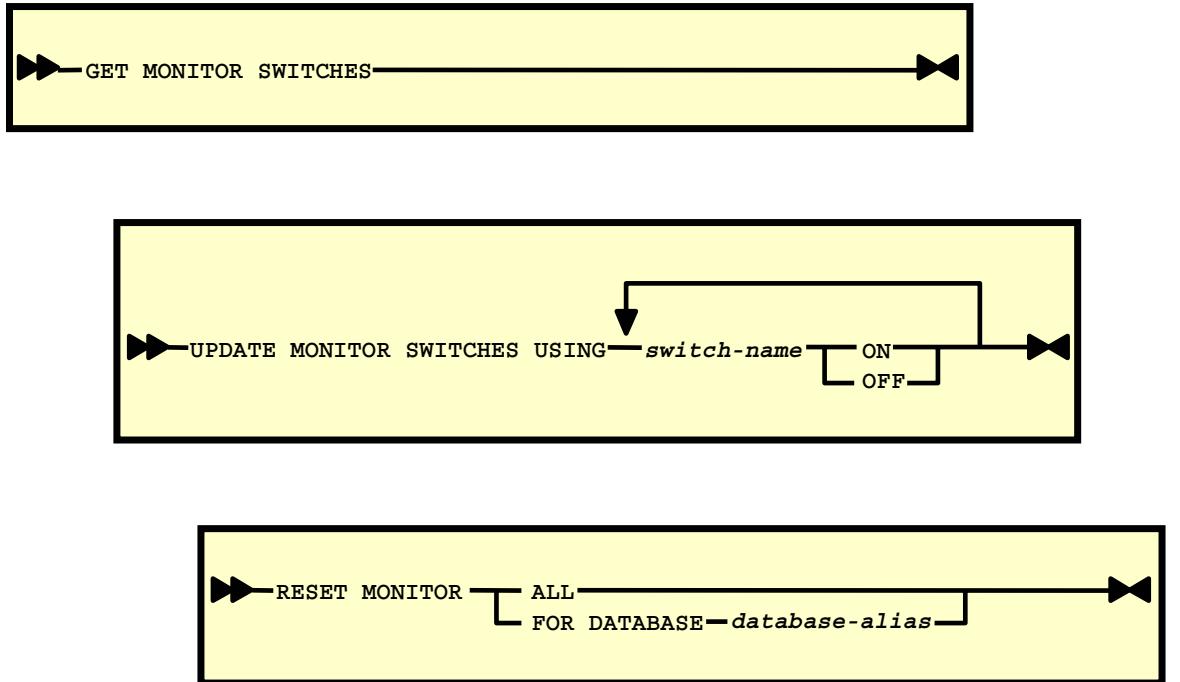
Notes:

Certain items are always monitored and available for analysis. These items are classified as **BASIC** monitor elements. Some items are only monitored and available if specifically requested. These items are classified within a **FUNCTIONAL GROUP**.

It is possible to monitor only BASIC information or to monitor BASIC information plus one or more FUNCTIONAL GROUPS. You should not monitor FUNCTIONAL GROUPS on a permanent basis due to system overhead of monitoring the elements within the group. The monitor architecture allows FUNCTIONAL GROUP details to be requested when necessary to perform certain analysis on the system.

For example, assume that an installation is experiencing long lock waits. The **LOCKS** functional group could be monitored during the time that the lock waits occur in an effort to resolve the problem. This functional group will provide, in addition to lock information considered basic, the agent holding the lock waited on, the object name, the object type, the table FID of the locked object, and the application holding the lock. Applications that are holding locks for long periods of time, and therefore contributors to lock wait problems, could be identified through these monitor elements.

Snapshot Monitor — CLP commands



© Copyright IBM Corporation 2007

Figure 9-13. Snapshot Monitor — CLP commands

CF238.3

Notes:

The commands listed above provide the capability to use the Database System Monitor via a Command Line Processor session.

GET MONITOR SWITCHES will return the current setting of the monitor functional groups illustrated previously. If a switch is ON, the date and time the group became active will also be displayed.

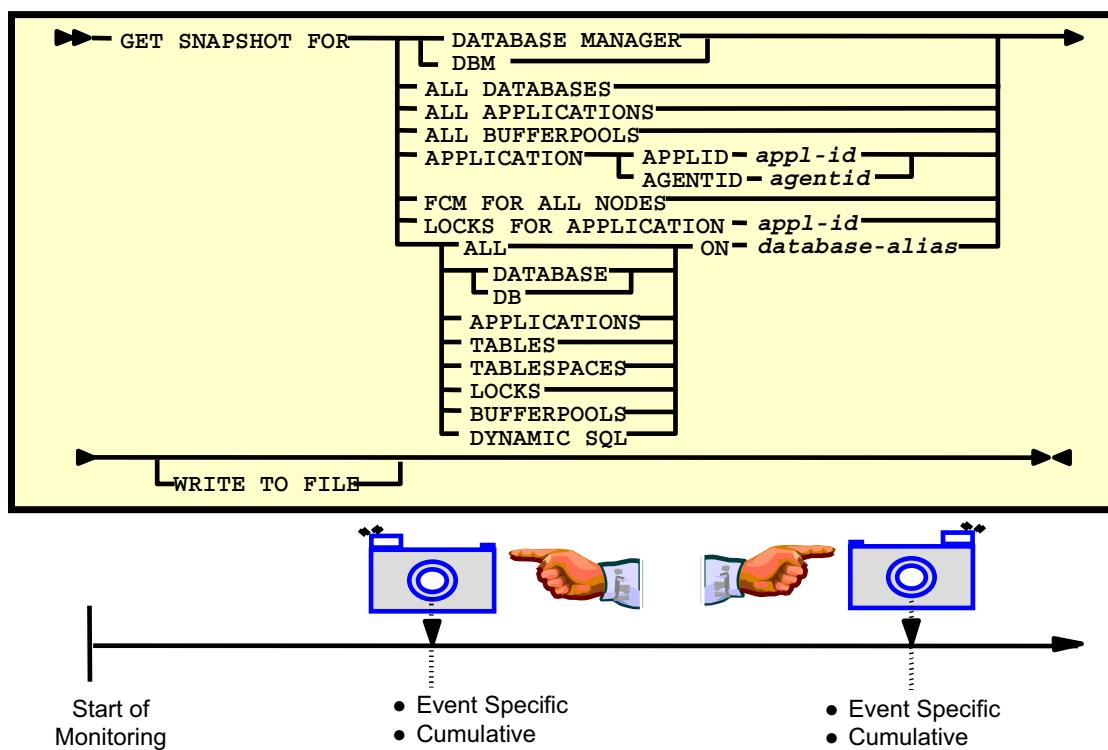
UPDATE MONITOR SWITCHES allows one or more functional groups to be turned on or off. The switch names that can be specified are as follows:

- BUFFERPOOL
- LOCK
- SORT
- STATEMENT
- TABLE
- TIMESTAMP
- UOW

Turning a functional group off and then on will cause all resettable monitor elements within the group to be reset to zero. These elements are counter-oriented. This reset impacts all databases defined to the instance for the application which issues the RESET statement.

RESET MONITOR will reset monitor elements to zero. The elements reset are counter-oriented. Note that this command can be directed for a specific database or for all databases defined to the instance. This command is NOT specific to a given monitor functional group.

Snapshot Monitor — taking snapshots



© Copyright IBM Corporation 2007

Figure 9-14. Snapshot Monitor — taking snapshots

CF238.3

Notes:

GET SNAPSHOT can be issued with a wide degree of scope concerning the level of information and the type of monitoring.

A snapshot, as the name implies, is a point-in-time view of the monitoring elements being analyzed. Some of these elements are cumulative in nature. They will contain information relative to the start of monitoring (or reset of counters) to the point of the snapshot. Other elements are related to specific events occurring within the system. These events may not appear on two consecutive snapshots, or they may contain totally different and unrelated information.

For example, assume that the following series of commands were issued during a CLP session.

- RESET MONITOR ALL
- GET SNAPSHOT FOR DATABASE ON musicdb
- (analyze information)
- GET SNAPSHOT FOR DATABASE ON musicdb
- (analyze information)

An element that exemplifies a cumulative counter that would be available in the snapshots taken would be **deadlocks detected**. This element would contain the number of deadlocks detected since the RESET MONITOR command. On the first snapshot, this element may be 0, while the second snapshot may indicate a higher number.

An element that exemplifies an event-specific piece of information that would be available in the snapshots taken would be **lock list memory in use**. This element would contain the number of bytes currently in use for the lock list. In an active system, it is unlikely that this element will have the exact same value from snapshot to snapshot. The number will fluctuate as applications obtain and release locks.

An alternative to the GET SNAPSHOT command is to issue a SELECT statement against table functions provided with DB2. Using the SQL SELECT interface allows you to select individual monitor elements from the returned data, as well as to use the filtering and aggregation capabilities of selects.

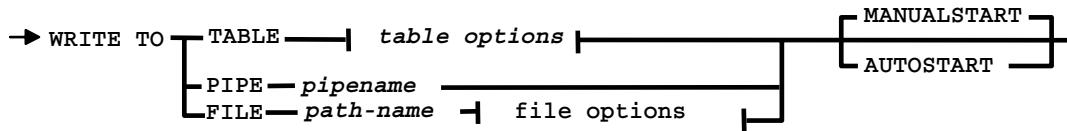
Event Monitor



Output of events:

- Table
- File
- Named pipe

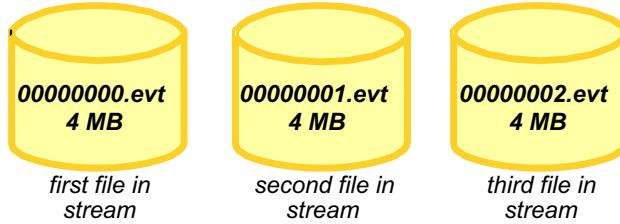
`CREATE EVENT MONITOR — event-monitor-name — FOR — event type` →



Example file structure

```
'/evmon1' directory
MAXFILESIZE 1024 (4 KB)
MAXFILES 3
```

Event Monitor turned OFF
after 3 files generated



© Copyright IBM Corporation 2007

Figure 9-15. Event Monitor

CF238.3

Notes:

A table event monitor streams event records to SQL tables, presenting a simple alternative to file and pipe event monitors in enabling you to easily capture, parse, and manage event monitoring data. For every event type an event monitor collects, target tables are created for each of the associated logical data groups.

If the event monitor output is directed to a file, the data is written into a number of files up to MAXFILES, where each file is of size MAXFILESIZE. Once the limit has been reached, the event monitor turns off, and logs that it did so in the administrative notification log file.

In the example above, the event monitor would log 12 MB of event data before turning itself off.

When an event monitor is created to write its output data into a file, there are a number of file options available. The creator must specify a directory location for the monitor files. The directory does not have to exist at CREATE EVENT MONITOR time. However, a check is made for the existence of the target path when the event monitor is activated. At that time, if the target path does not exist, an error is raised.

The options include:

MAXFILES	Specifies that there is a limit on the number of event monitor files that will exist for a particular event monitor at any time. The default is that there is no maximum (MAXFILES NONE).
MAXFILESIZE	Specifies that there is a limit to the size of each event monitor file. Default: (Windows) 200 4 KB pages; (UNIX) 1000 4 KB pages
BUFFERSIZE	Specifies the size of the event monitor buffers (4 KB pages). When the event monitor is started, two buffers of the specified size are allocated; these permit asynchronous I/O. The larger the buffers, the less I/O will be performed. The minimum and default size of each buffer is 4 pages (two buffers, each 16 KB in size). The maximum size of the buffers is limited by the size of the monitor heap (MON_HEAP_SZ), since the buffers are allocated from the heap. If you are using many event monitors at the same time, increase the size of the MON_HEAP_SZ DBM configuration parameter.
BLOCKED	Specifies that each agent that generates an event should wait for an event buffer to be written out of disk if the agent determines that both event buffers are full. BLOCKED should be selected to guarantee no event data loss. This is the DEFAULT.
NONBLOCKED	Specifies that each agent that generates an event should not wait for the event buffer to be written out to disk if the agent determines that both event buffers are full. NONBLOCKED event monitors do not slow down database operations to the extent of BLOCKED event monitors. However, NONBLOCKED event monitors are subject to data loss on highly active systems.
APPEND	Specifies that if event data files already exist when the event monitor is turned on, then the event monitor will append the new event data to the existing stream of data files. When the event monitor is reactivated, it will resume writing to the event files as if it had never been turned off. APPEND is the default option.
REPLACE	Specifies that if event data files already exist when the event monitor is turned on, then the event monitor will erase all of the event files and start writing data to file 00000000.evt.
MANUALSTART	Specifies that the event monitor should not be started automatically each time the database is started. Event monitors with the MANUALSTART option must be activated manually using the SET EVENT MONITOR STATE statement. This is the default.
AUTOSTART	Specifies that the event monitor be started automatically each time the database is started.

It is recommended that you send monitor output to a different volume or disk than the database or log files.

Filtering Event Monitor output

- Creator can specify filters using WHERE clause
- Example:

```
CREATE EVENT MONITOR georgepay FOR STATEMENTS  
WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'GEORGE'  
WRITE TO FILE '/monitors/georgepay'  
MAXFILES 25  
MAXFILESIZE 1024  
NONBLOCKED  
APPEND  
– Catch whenever GEORGE invokes payroll application  
– WHERE conditions can be specified for:  
    • Connection  
    • Statements  
    • Transactions
```

© Copyright IBM Corporation 2007

Figure 9-16. Filtering Event Monitor output

CF238.3

Notes:

Specific event information for connections, statements, or transactions may be captured based on APPL_ID, AUTH_ID, or APPL_NAME.

Remember that any number of event monitors can be defined (created), but only 32 can be active at a time.

To start using the Event Monitor, it must be started:

```
SET EVENT MONITOR georgepay STATE = 1
```

To stop using the Event Monitor:

```
SET EVENT MONITOR georgepay STATE = 0
```

Examining Event Monitor output



How do I examine Event Monitor output?

- Write an application to read the file or to receive pipe output
- Use *db2evmon* productivity tool (can write to a pipe or file only)
- Use the GUI event analyzer tool
- Use SQL against event monitor tables

© Copyright IBM Corporation 2007

Figure 9-17. Examining Event Monitor output

CF238.3

Notes:

Event monitor output can be viewed using the tool DB2EVMON, or an application can be written using the DB2 APIs to collect and display the monitor information. Data written to tables can be examined using SQL.

The DB2EVMON command formats event monitor file and named pipe output, and writes it to standard output. The command syntax is as follows:

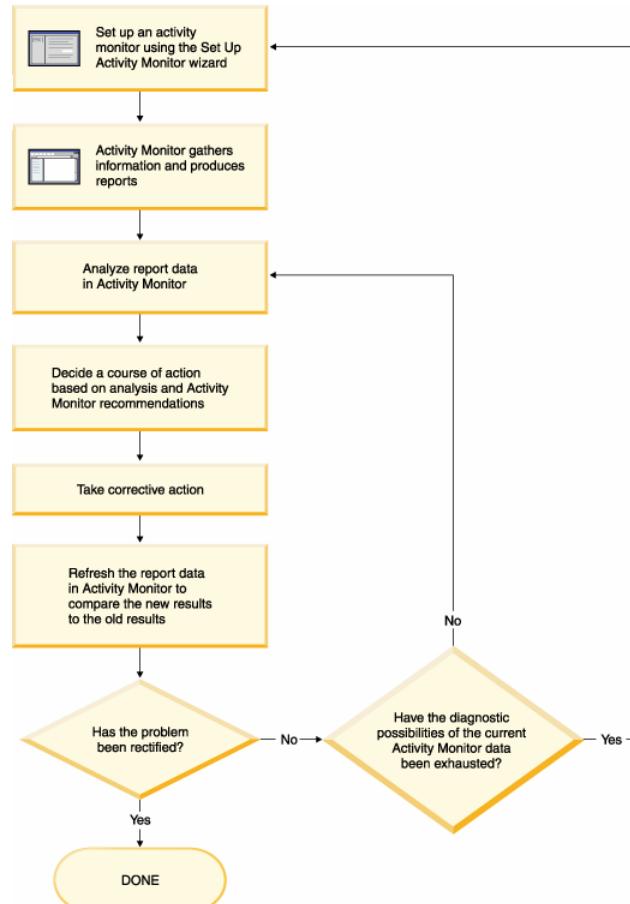
```
db2evmon database-alias event-monitor-name
```

If the event writes the data to a FILE, the db2evmon tool formats the files and writes the output to standard out.

If the event writes the data to a PIPE, the tool formats the output to standard out as the event occurs.

Activity Monitor

- Monitors a database or database partition for
 - Application performance and concurrency
 - Resource consumption
 - SQL statement usage
- Provides predefined reports based on a specific subset of monitor data
- Gives recommendations to
 - Assist in diagnosing the cause of database performance problems
 - Tune queries so that use of database resources is optimized



© Copyright IBM Corporation 2007

Figure 9-18. Activity Monitor

CF238.3

Notes:

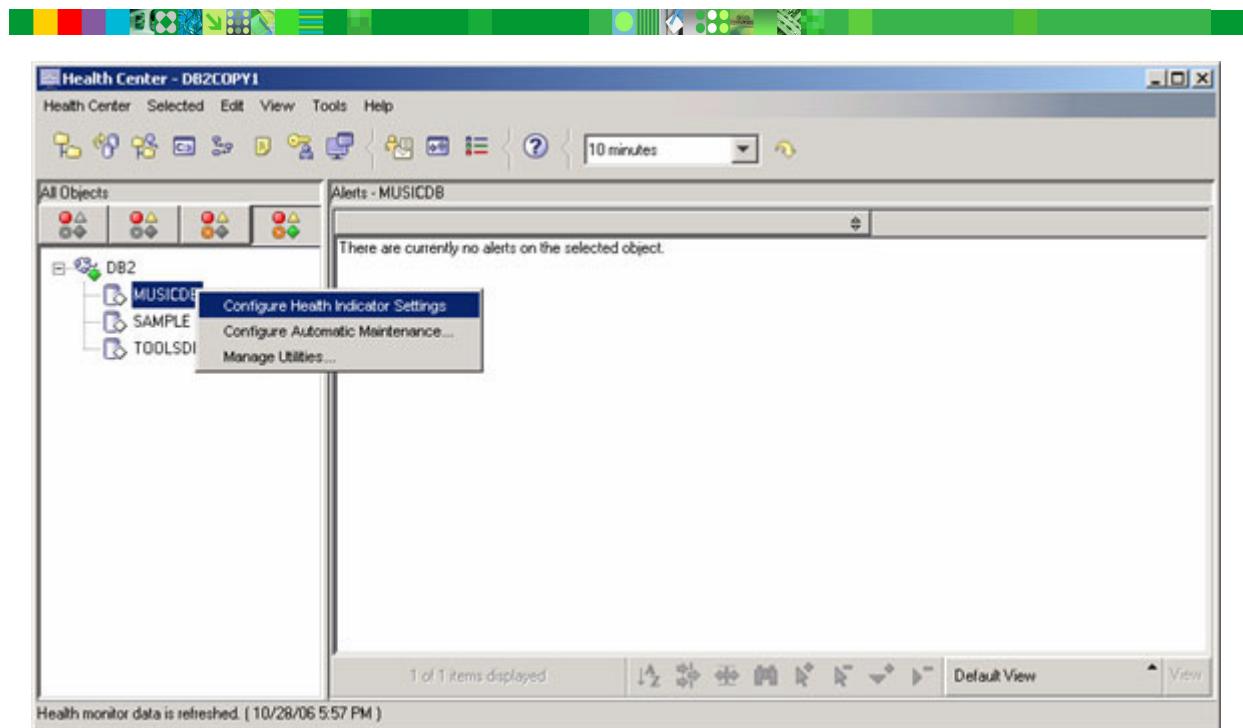
To access the Activity Monitor GUI from the Command Center:

- Expand a particular database from the explorer window (for example, SAMPLE or MUSICDB)
- Right-click and select **Activity Monitors --> Create**
- Choose what partition is to be monitored
- Then select or create a monitoring task (well explained in the GUI)

Tasks that you can perform from the Activity Monitor

<i>Tasks from the Activity Monitor</i>	<i>Aspects of tasks</i>	<i>Invocation</i>
Transactions	View transactions running on a selected application	Select one or more applications in the <i>Report data</i> pane. Right-click and select Show Latest Transactions . The <i>Application Transactions</i> window opens.
Statements	View SQL statements running on a selected application.	View SQL statements running on a selected application. Select one or more applications in the <i>Report data</i> pane. Right-click and select Show Latest Statements . The <i>Application Statements</i> window opens.
	View the text of SQL statements running on a selected application.	Select one or more applications in the <i>Report data</i> pane. Right-click and select Show Latest Statements . The <i>Application Statements</i> window opens.
Application Lock Chains	<p>View locks and lock-waiting situations that currently affect a selected application.</p> <p>View information about a selected application for which you are viewing lock information.</p> <p>View information about the locks held and the locks waited on by a selected application in your database.</p>	<p>Select an application in the <i>Report data</i> pane. Right-click and select Show Lock Chains. The <i>Application Lock Chains</i> window opens.</p> <p>From the <i>Application Lock Chains</i> window, right-click an application, and select About.</p> <p>From the <i>Application Lock Chains</i> window, right-click an application, and select Show Lock Details.</p>
View report data and recommendations	<p>View information to help you interpret report data.</p> <p>View recommendations provided by Activity Monitor.</p>	<p>From an <i>Activity Monitor</i> window, an <i>Application Statements</i> window, or an <i>Application Transactions</i> window, use the <i>Report arrow</i> to select the report and click the Report Details push button. View the <i>Details</i> page.</p> <p>From an <i>Activity Monitor</i> window, an <i>Application Statements</i> window, or an <i>Application Transactions</i> window, use the <i>Report arrow</i> to select the report and click the Report Details push button. View the <i>Recommendations</i> page.</p>

Health Monitor



```
db2 get alert cfg for database on musicdb
```

© Copyright IBM Corporation 2007

Figure 9-19. Health Monitor

CF238.3

Notes:

The Health Monitor is a server-side tool that constantly monitors the health of the instance, even without user interaction. If the Health Monitor finds that a defined threshold has been exceeded (for example, the available log space is not sufficient), or if it detects an abnormal state for an object (for example, an instance is down), the Health Monitor will raise an alert.

A *health indicator* is a system characteristic that the Health Monitor checks. The Health Monitor comes with a set of predefined thresholds for these health indicators. The Health Monitor checks the state of your system against these health-indicator thresholds when determining whether to issue an alert. Using the Health Center, commands, or APIs, you can customize the threshold settings of the health indicators, and define who should be notified and what script or task should be run if an alert is issued.

The Health Center provides the graphical interface to the Health Monitor. You use it to configure the Health Monitor, and to configure Automatic Maintenance, as well as to manage utilities and to see the rolled-up alert state of your instances and database objects.

The Health Center and Control Center are integrated through Health Beacons. Health Beacons in the Control Center provide notifications about new alerts in the Health Center. Beacons are implemented on all Control Center windows and notebooks; simply click on a Health Beacon to access the Health Center.

The Health Monitor gathers information about the health of the system using interfaces that do not impose a performance penalty. It does not turn on any snapshot monitor switches to collect information. The Health Monitor is enabled by default when a instance is created; you can deactivate it using the database manager configuration parameter *health_mon*.

DB2 UDB also provides a Web Health Center that can be used to access the Health Monitor information from a Web browser or PDA.

You can also use DB2 commands and APIs to retrieve health information from the Health Monitor, allowing you to integrate DB2 Health Monitoring with existing system-wide monitoring solutions. For example, to obtain the current configuration settings for an instance, database, table space, or container, you can use the GET ALERT CFG command. For example:

```
db2 get alert cfg for database on musicdb using db.spilled_sorts
```

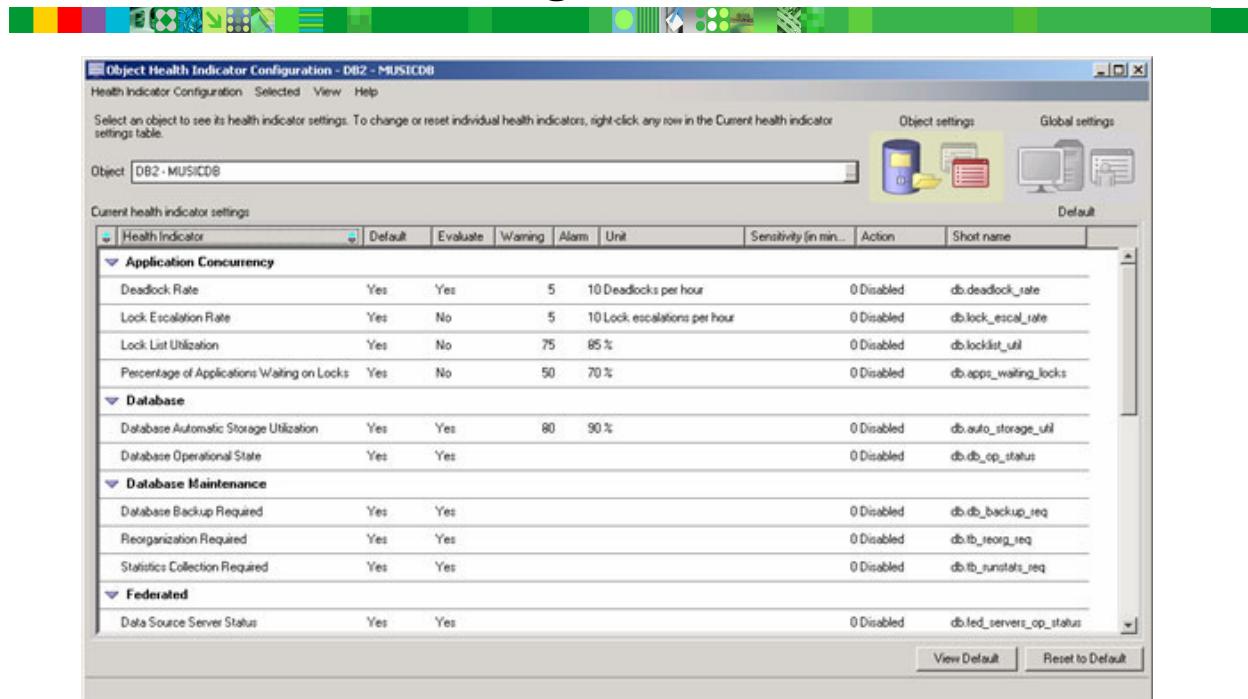
Alert Configuration

Indicator Name	= db.spilled_sorts
Type	= Threshold-based
Warning	= 30
Alarm	= 50
Sensitivity	= 0
Formula	= (db.sort_overflows/db.total_sorts)*100;
Actions	= Disabled
Threshold or State checking	= Enabled

```
SQL22004N Cannot find the requested configuration for the given object.  
Returning default configuration for "databases".
```

Note that for this example, the database SAMPLE did not have specific settings; it was using the default settings for all databases.

Health indicator settings



```
db2 get description for health indicator db.spilled_sorts
db2 update alert cfg for database on musicdb using
  db.spilled_sorts set alarm 50, warning 30
```

© Copyright IBM Corporation 2007

Figure 9-20. Health indicator settings

CF238.3

Notes:

Health indicators are used by the Health Monitor to evaluate specific aspects of database manager or database performance. A health indicator is a specific measurement that gauges the healthiness of some aspect of a particular class of database objects, for example table spaces. Health indicators measure either a finite set of distinct states or a continuous range of values to determine whether the state is healthy or unhealthy. The state defines whether or not the database object or resource is operating normally. If the change in the state is determined to be unhealthy, the Health Monitor issues an alert through the specified reporting channels.

An alert is generated in response to either a change to a non-normal state or the value of the health indicator falling into a warning or alarm zone based on defined threshold boundaries. For health indicators measuring distinct states, an attention type alert is issued if a non-normal state is registered. For health indicators measuring a continuous range of values, threshold values define boundaries or zones for normal, warning, and alarm states. If, for example, the value enters the threshold range of values that defines an alarm zone, an alarm type alert is issued to indicate that the problem needs immediate attention. There are three types of alerts: alarm, warning, and attention.

Health Monitor information can be accessed through the Health Center, Web Health Center, the CLP, or APIs. Health indicator configuration is available through these same tools.

The following are the categories of health indicators:

- Table space storage
- Sorting
- Database management system
- Database
- Logging
- Application concurrency
- Package and catalog caches, and workspaces
- Memory

Command line examples

```
db2 get alert cfg for database on musicdb using db.spilled_sorts
```

```
Alert Configuration
```

Indicator Name	= db.spilled_sorts
Type	= Threshold-based
Warning	= 30
Alarm	= 50
Sensitivity	= 0
Formula	= (db.sort_overflows/db.total_sorts)*100;
Actions	= Disabled
Threshold or State checking	= Enabled

SQL22004N Cannot find the requested configuration for the given object.
Returning default configuration for "databases".

```
db2 get description for health indicator db.spilled_sorts
```

```
DESCRIPTION FOR db.spilled_sorts
```

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily. Sorts that overflow to disk can cause significant performance degradation. If this occurs, an alert may be generated. The indicator is calculated using the formula: (db.sort_overflows /db.total_sorts)*100. The system monitor data element db.sort_overflows is the total number of sorts that ran out of sort heap and may have required disk space for temporary storage. The data element db.total_sorts is the total number of sorts that have been executed.

```
db2 update alert cfg for database on musicdb using db.spilled_sorts set alarm 60, warning 40  
DB20000I The UPDATE ALERT CONFIGURATION command completed successfully.
```

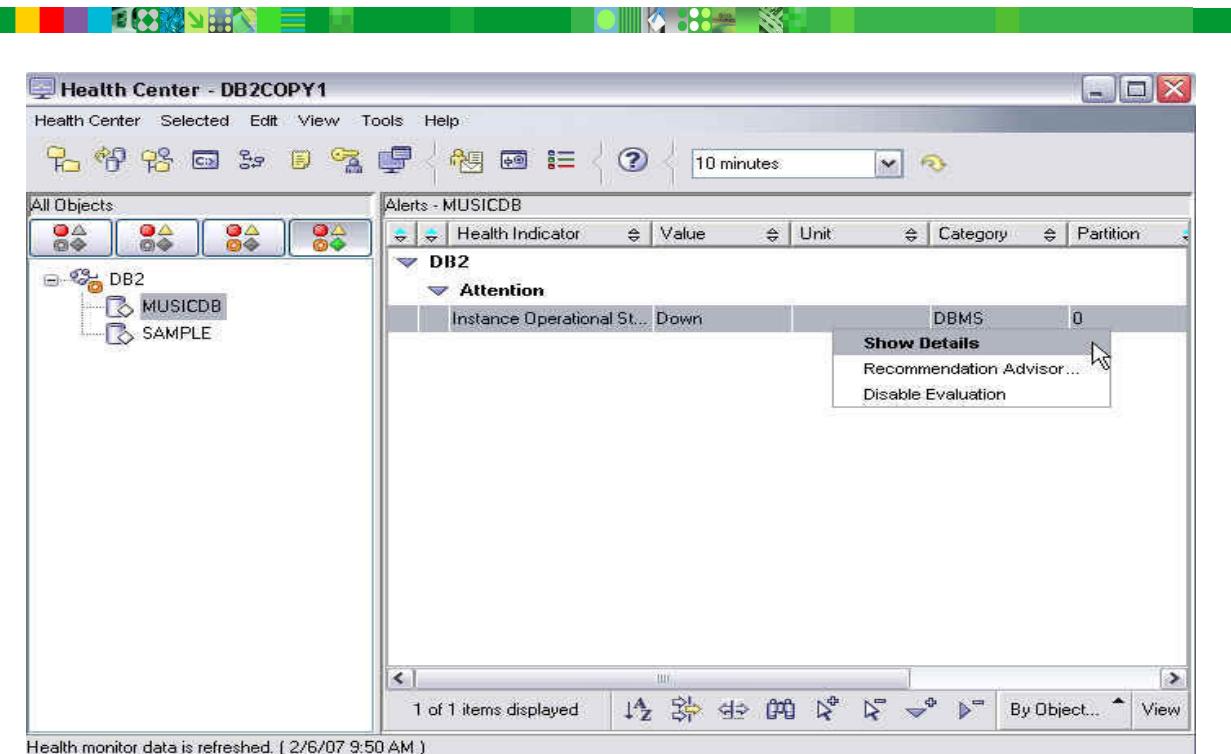
```
db2 get alert cfg for database on musicdb using db.spilled_sorts
```

```
Alert Configuration
```

Indicator Name	= db.spilled_sorts
Type	= Threshold-based
Warning	= 40
Alarm	= 60
Sensitivity	= 0
Formula	= (db.sort_overflows/db.total_sorts)*100;
Actions	= Disabled
Threshold or State checking	= Enabled

Notice that, compared with example #1, there is now a specific setting for the MUSICDB database, so the SQL22004 message is no longer returned.

Health Monitor — alerts view



```
db2 get health snapshot for database manager
```

© Copyright IBM Corporation 2007

Figure 9-21. Health Monitor — alerts view

CF238.3

Notes:

When an alert is raised, two things can occur:

- Alert notifications can be sent by e-mail or to a pager address, allowing you to contact whoever is responsible for a system.
- Preconfigured actions can be taken. For example, a script or a task (implemented from the new Task Center) can be run.

Using the Health Monitor's drill-down capability, you can access details about current alerts and obtain a list of recommended actions that describe how to resolve the alert.

db2 get health snapshot for database manager

```
Database Manager Health Snapshot

Node name = 
Node type = Enterprise Server Edition with local and remote clients
Instance name = DB2
Snapshot timestamp = 10/30/2006 17:59:32.826677

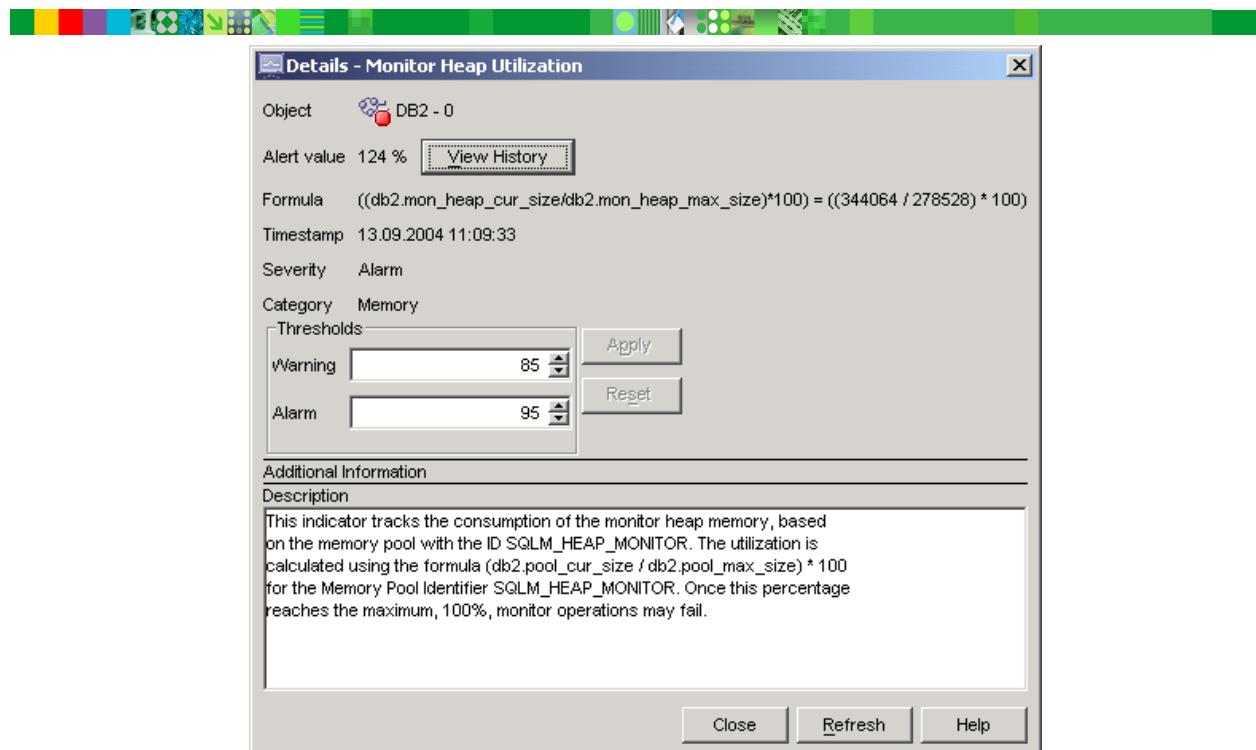
Number of database partitions in DB2 instance = 1
Start Database Manager timestamp = 10/23/2006 11:59:14.567027
Instance highest severity alert state = Alarm

Health Indicators:

Indicator Name = db2.db2_op_status
Value = 0
Evaluation timestamp = 10/30/2006 17:55:30.090000
Alert state = Normal

Indicator Name = db2.mon_heap_util
Value = 100
Unit = %
Evaluation timestamp = 10/30/2006 17:55:30.090000
Alert state = Alarm
```

Alarm details



db2 get recommendations for health indicator db2.mon_heap_util

© Copyright IBM Corporation 2007

Figure 9-22. Alarm details

CF238.3

Notes:

You can follow one of the recommended actions to address the alert. If the recommended action is to make a database or database manager configuration change, a new value will be recommended, and you can implement the recommendation by clicking a button. In other cases, the recommendation will be to investigate the problem further by launching a tool, such as the CLP or the Memory Visualizer.

db2 get recommendations for health indicator db2.mon_heap_util

Recommendations:

Recommendation: Investigate memory usage of monitor heap.

This parameter determines the amount of the memory, in pages, to allocate for database system monitor data. Memory is allocated from the monitor heap for database monitoring activities such as taking a snapshot, turning on a monitor switch, resetting a monitor, or activating an event monitor.

For more information on the monitor heap, refer to the DB2 Information Center.

Investigate the amount of memory that was used for the monitor heap over time to determine the most appropriate value for the monitor heap configuration parameter. The database system monitor tracks the highest amount of memory that was used for the monitor heap.

Take one of the following actions:

Launch DB2 tool: Memory Visualizer

The Memory Visualizer is used to monitor memory allocation within a DB2 instance. It can be used to monitor overall memory usage, and to update configuration parameters for individual memory components.

To open the Memory Visualizer:

1. From the Control Center, expand the object tree until you find the instances folder.
2. Click the instances folder. Any existing instances are displayed in the contents pane on the right side of the window.
3. Right-click the instance that you want in the Contents pane, and click View Memory Usage in the pop-up menu. The Memory Visualizer opens.

To start the Memory Visualizer from the command line issue the db2memvis command.

The Memory Visualizer displays a hierarchical list of memory pools for the database manager. Monitor Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.

Click the check box on the Show Plot column for the Monitor Heap row to add the element to the plot.

Recommendation: Increase the monitor heap size.

Increase the database manager configuration parameter mon_heap_sz

sufficiently to move utilization to normal operating levels. Set the new value of mon_heap_sz to be equal to (pool_cur_size / (4096*U)) where U is the desired utilization rate. For example, if the desired utilization rate is 60% of the warning threshold level, which has been set at 75%, then $U = 0.6 * 0.75 = 0.45$ (or 45%).

Take one of the following actions:

Launch DB2 tool: DBM Configuration Window

The DBM Configuration window can be used to view and update database manager configuration parameters.

To open the DBM Configuration window:

1. From the Control Center, expand the object tree until you find the instances folder.
2. Click the instances folder. Any existing instances are displayed in the contents pane on the right side of the window.
3. Right-click the instance that you want in the Contents pane, and click Configure Parameters in the pop-up menu. The DBM Configuration window opens.

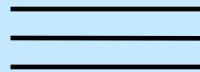
Expand the Performance category, then update the monitor heap size parameter as suggested and click OK to apply the update.

From the Command Line Processor, issue the commands shown in the following example:

```
UPDATE DATABASE MANAGER CONFIGURATION USING MON_HEAP_SZ size
```


Independent trace facility — *db2trc*

db2trc on



db2trc dump trace.bin

db2trc off

db2trc fmt trace.bin trace.fmt

edit trace_fmt

(Windows)

vi trace.fmt

(UNIX)

trace.fmt

```
DB2 cei_retcode    base_sys_utilities    sqlegsem (1.23.5.97)
pid 17027; fid 1; cpid 5; time 0; trace_point 254
return_code = 000000 = 0

DB2 cei_entry      oper_system_services sqlcinca (1.20.151.121)
pid 17027; fid 1; cpid 5; time 0; trace_point 0
```

© Copyright IBM Corporation 2007

Figure 9-23. Independent trace facility — *db2trc*

CF238.3

Notes:

DB2 provides the *db2trc* command to trace DB2 events. The *db2trc* command collects event information, dumps the trace to a file, and formats the trace dump file.

db2trc should be performed with the assistance of your IBM Support Center. SYSADM, SYSCTRL, or SYSMAINT authority is needed to run *db2trc*.

Running the *db2trc* command:

- Start the trace.

db2trc on

Perform the action you wish to trace.

- Dump the trace to a file. If no directory path is specified, then the trace file will be written to the current directory.

db2trc dump trace.bin

- Turn the trace off.

```
db2trc off
```

- Format the dump trace file. The `fmt` option lists each event chronologically. The `flw` option sorts each event by process or thread. If the output is not redirected, then the output will go to the terminal.

```
db2trc fmt trace.bin trace.format
```

```
db2trc flw trace.bin trace.flow
```

- Send the formatted trace to the IBM Support Center.

`db2trc on -l 10000000` sets aside a 10 MB buffer for the trace. Here `-l` is the letter *e*/— not the digit *one*.

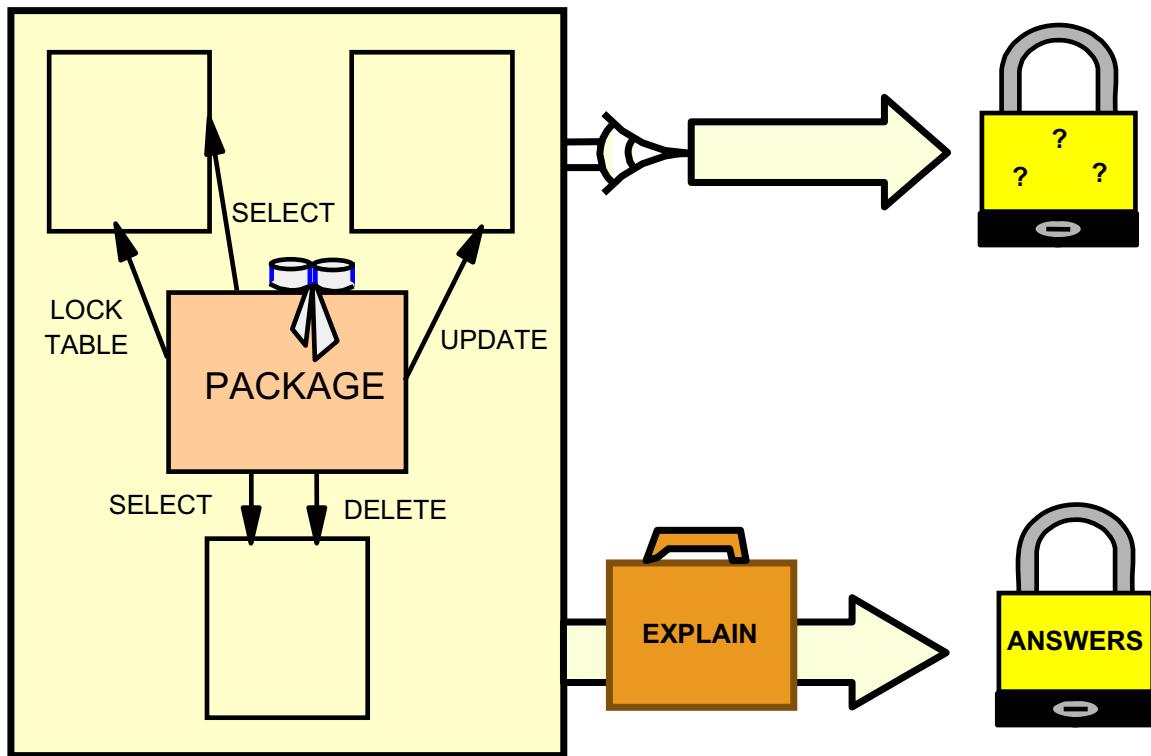
The following are trace options:

- f *filename*** Specifies that trace information should be continuously written to the specified file, until db2trc is turned off.
- p *pid.tid*** Only enables the trace facility for the specified process IDs (pid) and thread IDs (tid). A period must be included if a tid is specified.
- l *buffer_size*** Specifies retention of the last trace records (the first records are overwritten when buffer is full)
- i *buffer_size*** Specifies retention of the initial trace records (no more records are written to the buffer once it is full)

Refer to the *Command Reference* for further information.

9.3 EXPLAIN

Explain tool



© Copyright IBM Corporation 2007

Figure 9-24. Explain tool

CF238.3

Notes:

Analyzing an application by viewing a listing may provide insight regarding the application's function and flow. However, the manner in which SQL will be executed is externalized via EXPLAIN.

Questions that could be answered through the use of EXPLAIN include:

- What indexes are used for this statement?
- What is the order of table access?
- What are the locking requirements for this program?
- Is any sorting required, and is it planned to be piped?

Consult the *Administration Guide* for additional details.

EXPLAIN facility tools



Requirements	Visual Explain	Explain Tables	db2exfmt	db2expln
GUI interface	✓			
Text output			✓	✓
"Quick and dirty" static SQL analysis				✓
Static SQL supported	✓	✓	✓	✓
Dynamic SQL supported	✓	✓	✓	✓*
CLI applications supported	✓	✓	✓	
Available to DRDA application requesters		✓		
Detailed optimizer information	✓	✓	✓	
Suited for analysis of multiple statements		✓	✓	✓
Information available from within application		✓		

Note: * Creates static package and explains it

© Copyright IBM Corporation 2007

Figure 9-25. EXPLAIN facility tools

CF238.3

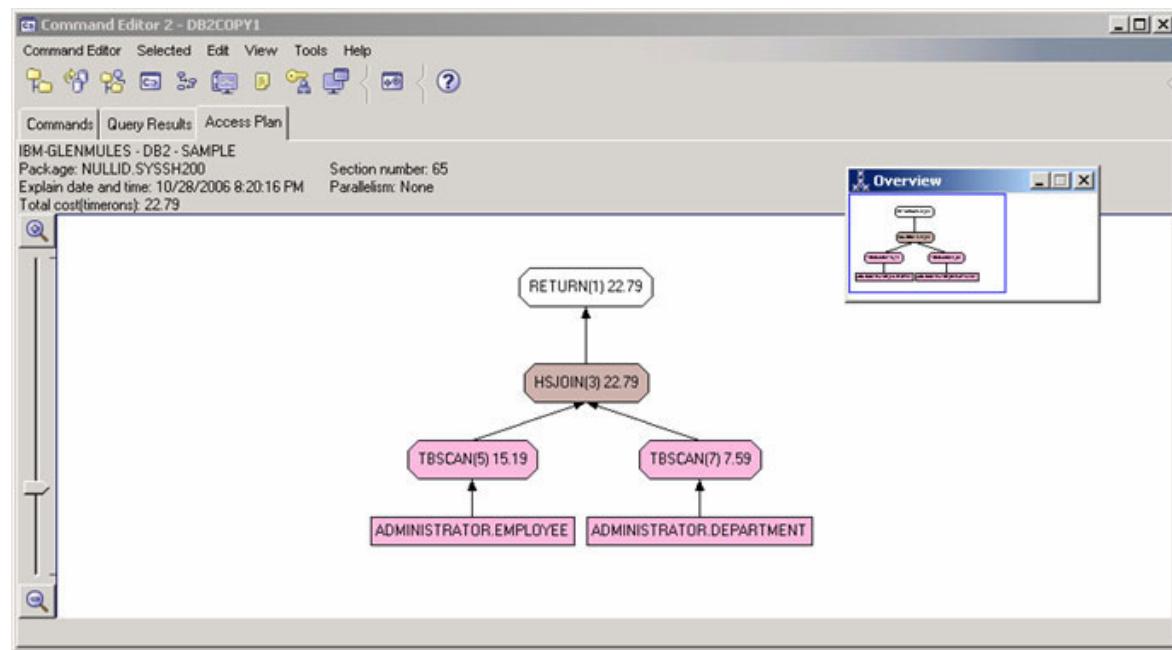
Notes:

The table summarizes the different tools available within the DB2 EXPLAIN facility and their individual characteristics. Use this table to select the tool most suitable for your environment and needs.

- Visual Explain allows for the analysis of access plan and optimizer information from the EXPLAIN tables through a graphical interface.
- The EXPLAIN Tables are accessible on all supported platforms and can contain information for both static and dynamic SQL statements. You can access the EXPLAIN tables using SQL statements, which allow for easy manipulation of the output and comparisons of the same query over time.
- db2exfmt allows you to obtain reports from the explain tables in a predefined format.
- db2expln will allow you to see the access plan information that is stored in the system catalog as part of a static package. It provides a text-based report on the strategy DB2 will use to execute the statements.

More details on the above EXPLAIN tools can be found in your *Administration Guide*.

Visual explain access plan graph



```
SELECT *
FROM administrator.employee e, administrator.dept d
WHERE e.workdept = d.deptno
```

© Copyright IBM Corporation 2007

Figure 9-26. Visual explain access plan graph

CF238.3

Notes:

The statement being explained on the graphic is:

```
SELECT *
FROM administrator.employee e, administrator.dept d
WHERE e.workdept = d.deptno
```

To access the Explain SQL Statement (Visual Explain) GUI:

- Right-click and select **Database**
 - Left-click and select **Explain SQL**

From the Explain SQL Statement GUI, an administrator can insert SQL text, save text, set a query tag, and set the optimization level.

Visual Explain allows for the analysis of access plan and optimizer information from the explain tables through a graphical interface. Both static and dynamic SQL statements can be analyzed using the tool. The analyzed statement can be either a previously bound static package or a previously explained dynamic SQL statement. Visual Explain is typically invoked from within the Command Editor.

There are two methods to gain access to Visual Explain from the Control Center:

- Explain statement history --> Open as details
- Packages --> (select the desired package) --> Open as Explain statement history

Visual Explain can also be accessed in the Command Center by identifying in the Command Center options that you want to automatically generate access plans, and then executing an explainable statement in the interactive page of the Command Center.

Visual Explain allows you to graph snapshots captured on another platform. For example, a Windows client can graph snapshots generated on a DB2 for Solaris server.

9.4 Additional commands

Additional commands



- **db2 ? command**
 - Request help text associated with command
- **db2 ? SQLnnnnN**
 - Requests help for a message specified by a valid SQLCODE nnnn
 - N = negative (most SQLCODES are negative), W = warning
- **db2 DESCRIBE TABLE tablename**
 - Displays columns of a table or a view
 - Optional SHOW DETAIL clause
- **db2 DESCRIBE INDEXES FOR TABLE tablename**
 - Displays indexes for a table
 - Optional SHOW DETAIL clause
- **Other uses for the DESCRIBE command**

© Copyright IBM Corporation 2007

Figure 9-27. Additional commands

CF238.3

Notes:

db2 ? command requests the help text associated with the command given. Note that this only provides help for commands, not for SQL statements.

db2 ? SQLnnnnN provides help for the message specified. At least a 4-digit number must be specified. The final letter (often “N” or “W” is optional), as the meaning is determined by the number.

db2 describe table *table-name* displays the columns of a table or a view.

db2 describe table *table-name* show detail gives you additional information.

db2 describe indexes for table *table-name* displays the indexes for a table.

You can apply **db2 ? command** to the DESCRIBE command itself. This will give you some additional ideas on what can be described:

```
DESCRIBE { [OUTPUT] {select-statement | call-statement | XQUERY xquery-statement} |
           {TABLE | {INDEXES | DATA PARTITIONS} FOR TABLE} table-name [SHOW DETAIL]}
```


LIST ACTIVE DATABASES command



```
db2 LIST ACTIVE DATABASES
```

```
Database name          = MUSICDB
Applications connected currently = 4
Database path         = /home/inst00/inst00/NODE0000/SQL00001/
```

© Copyright IBM Corporation 2007

Figure 9-28. LIST ACTIVE DATABASES command

CF238.3

Notes:

The LIST ACTIVE DATABASES command can be used to display the active databases within an instance. The current number of active connections is also displayed for each database listed.

One of the following authorities is needed to run the LIST APPLICATIONS command:

- SYSADM
- SYSCTRL
- SYSMAINT
- SYSMON

LIST APPLICATIONS command



```
db2 LIST APPLICATIONS FOR DATABASE db_alias SHOW DETAIL
```

Auth Id	Application Name	Appl. Handle	Application Id
INST00	db2bp_32	78	*LOCAL.inst00.000327220543

Seq#	Number of Agents	Coordinating Node Number	Coordinating pid/thread	Status	Status Change Time
0001	1	0	230	UOW Waiting	Not Collected

Node	DBName	DB Path
0	MUSICDB	/home/inst00/inst00/NODE0000/SQL00001/

© Copyright IBM Corporation 2007

Figure 9-29. LIST APPLICATIONS command

CF238.3

Notes:

The LIST APPLICATIONS command displays an entry for each application connected to a specific database or to all databases within the instance (the default).

The output shows the application program name, authorization ID (user name), application handle, application ID, and the database name for each database application.

If the SHOW DETAIL parameter is used, the output will also display the application's sequence number, status, status change time, node, and database path.

One of the following authorities is needed to run the LIST APPLICATIONS command:

- SYSADM
- SYSCTRL
- SYSMAINT
- SYSMON

FORCE APPLICATION command



```
db2 FORCE APPLICATION { ALL | (application-handle) }
```

- User application handle from LIST APPLICATIONS
- Rolls back uncommitted transactions

```
db2stop force
```

© Copyright IBM Corporation 2007

Figure 9-30. FORCE APPLICATION command

CF238.3

Notes:

The FORCE APPLICATION command can be used to break the database connection between an application and its agent (db2agent). The FORCE APPLICATION command can be used to break the connections for ALL applications or specific applications.

When specifying specific applications, use the application handle number from the LIST APPLICATIONS command.

If an application is forced, an uncommitted Unit of Work will be rolled back. The force takes effect immediately. However, the command runs asynchronously and the user may regain control before all applications have been forced.

The FORCE command breaks the connection at the database server and does not terminate the database application. Do NOT use operating system commands to stop or kill a database agent. SYSADM, SYSCTRL, or SYSMON authority level is required to issue the FORCE APPLICATION command.

**Note**

Applications may also be forced while stopping the instance using the FORCE option on the db2stop command.

The db2set command

- Command line tool for DB2 Profile Registry administration
- Displays, sets, resets, or removes profile variables

```
db2set variable=value  
-g  
-i instance [db-partition-number]  
-n DAS node [[-u user id] [-p password]]  
-u  
-ul  
-ur  
-p  
-r  
-l  
-lr  
-v  
-? (or -h)  
-all
```

© Copyright IBM Corporation 2007

Figure 9-31. The *db2set* command

CF238.3

Notes:

The db2set command allows us to display, set (or reset), and remove various profile variables.

Some of these settings are important to diagnostic analysis and hence we are reviewing this command here.

Retrieve stats from a running instance: *db2pd*



db2pd (DB2 performance data)

- Allows you to retrieve statistics from a running DB2 instance or database
- Provides information useful for troubleshooting and problem determination, performance improvement, and application development design:
 - Locks / bufferpools / table spaces / containers / dynamic SQL statements / agents / applications / memory pools and sets / transactions / logs / ...
- Collects data without acquiring any latches and without using any engine resources — thus the data is expected to be constantly changing (and hence not completely accurate)

© Copyright IBM Corporation 2007

Figure 9-32. Retrieve stats from a running instance: *db2pd*

CF238.3

Notes:

db2pd is a new utility that can be used to retrieve statistics from a running DB2 instance or database.

The tool can provide a wide range of information useful for troubleshooting and problem determination, performance improvements, and application development design, including:

- locks
- buffer pools
- table spaces
- containers
- dynamic SQL statements
- agents
- applications
- memory pools and sets
- transactions
- logs

The tool collects this data without acquiring any latches and without using any engine resources. It is therefore possible (and expected) to retrieve data that is constantly changing while **db2pd** is collecting it — thus this data may not be completely accurate or unchanging (static) at the particular instance in time when the report is complete.

The primary benefit is the ability to collect and analyze data from a running instance or database without acquiring latches. Accordingly, we have fast retrieval and no competition for engine resources.

The command is modelled in concept after the **onstat** utility used with IBM Informix instances.

In a number of the options, **file=<filename>** is used to specify an output file name for the results of the command. A similar effect can be achieved by using output file redirection (**stdout** or **>**). Note that no spaces are generally allowed within **file=<filename>**.

Syntax (using db2pd -h)

Usage:

```
-h | -help [file=<filename>]
    Help
-v | -version [file=<filename>]
    Version
-osinfo [disk] [file=<filename>]
    Operating System Information
-dbpartitionnum <num>[,<num>]
    Database Partition Number(s)
-alldbpartitionnums
    All partition numbers
-database | -db <database>[,<database>]
    Database(s)
-alldatabases | -all dbs
    All Active Databases
-inst
    Instance scope output
-file <filename>
    All Output to Filename
-command <filename>
    Read in predefined options
-interactive
    Interactive
-full
    Expand output to full length
-repeat [num sec] [count]
    Repeat every num seconds (default 5) count times
-everything
    All options on all database partitions
```

Instance scope options:

```
agents [db=<database>] [ [agent=<agentid>] | [application=<appid>] ]
    [file=<filename>]
    Agents
-fcm [file=<filename>]
    FCM Information
```

```

-mempools [file=<filename>]
    Memory Pools
-memsets [file=<filename>]
    Memory Sets
-dbmcfg [file=<filename>]
    DBM Config
-sysplex [db=<database>] [file=<filename>]
    Sysplex List
-utilities [file=<filename>]
    Utilities

```

Database scope options:

```

-applications [ [application=<appid>] | [agent=<agentid>] ] [file=<filename>]
    Applications
-transactions [tran=<tranhdl>] [app=<apphdl>] [file=<filename>]
    Transactions
-bufferpools [file=<filename>]
    Buffer Pools
-logs [file=<filename>]
    Transaction Logs
-locks [tran=<tranhdl>] [file=<filename>] [showlocks]
    Locks
-tablespaces [file=<filename>] [group] [tablespace=<tablespace id>]
    Tablespaces/Containers
-dynamic [file=<filename>]
    Dynamic Cache
-static [file=<filename>]
    Static Cache
-mempools [file=<filename>]
    Memory Pools
-memsets [file=<filename>]
    Memory Sets
-dbcfg [file=<filename>]
    Database Config
-catalogcache [file=<filename>]
    Catalog Cache
-tcbstats [all|index] [tbspaceid=<tbspaceid> [tableid=<tableid>]] [file=<filename>]
    Table Control Block Stats
-reorg [file=<filename>]
    Table Reorg Stats
-recovery [file=<filename>]
    Recovery Status
-reopt [file=<filename>]
    Reoptimized SQL Statements

```

Examples:

```

db2pd -dbpartitionnum 0,1 -db sample -locks
db2pd -dbp 0,1 -database sample -locks app=<appid>
db2pd -alldbp -alldbs

```

Some Sample Reports

Operating System Information (db2pd -osinfo):

```
OSName: WIN32_NT
NodeName: IBM-VC5QJ67OIJYZ
Version: 5.0
Release: Service Pack 4
Machine: x86 Family 6, model 8, stepping 6
```

CPU Information:

TotalCPU	OnlineCPU	ConfigCPU	Speed (MHz)	HMTDegree
1	1	1	796	1

Physical Memory and Swap (Megabytes):

TotalMem	FreeMem	AvailMem	TotalSwap	FreeSwap
256	9	9	2177	1340

Virtual Memory (Megabytes):

Total	Reserved	Available	Free
2433	n/a	n/a	1349

Dynamic SQL statements held in server cache (db2pd -dynamic):

```
F:\IBM\DB2\SQLLIB\BIN>db2 connect to sample
```

Database Connection Information

```
Database server      = DB2/NT 9.1.0
SQL authorization ID = ADMINIST...
Local database alias = SAMPLE
```

```
F:\IBM\DB2\SQLLIB\BIN>db2 "SELECT * FROM employee"
```

EMPNO	FIRSTNAME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL
SEX	BIRTHDAY	SALARY						
-----	-----	-----	-----	-----	-----	-----	-----	-----
000010	CHRISTINE	I	HAAS	A00	3978	01/01/1965	PRES	18
F	08/24/1	4220.00						
000020	MICHAEL	L	THOMPSON	B01	3476	10/10/1973	MANAGER	18
M	02/02/1	3300.00						
000030	SALLY	A	KWAN	C01	4738	04/05/1975	MANAGER	20
F	05/11/1	3060.00						
...								

```
F:\IBM\DB2\SQLLIB\BIN>db2pd -dynamic -db sample
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:02:59
```

Dynamic Cache:

Current Memory Used	217553
Total Heap Size	1271398
Cache Overflow Flag	0
Number of References	2
Number of Statement Inserts	3
Number of Statement Deletes	0
Number of Variation Inserts	2

Number of Statements 3

Dynamic SQL Statements:

Address	AnchID	StmtID	NumEnv	NumVar	NumRef	NumExe	Text
0x034B05D0	101	1	0	0	0	0	SET CURRENT LOCALE LC_CTYPE 'en_US'
0x034B0830	144	1	1	1	1	1	SELECT * FROM employee
0x034B6A20	170	1	1	1	1	1	SELECT COUNT(*) FROM SYSCAT.PROCEDURES WHERE PROCNAME='SYSINSTALLROUTINES'

Dynamic SQL Environments:

Address	AnchID	StmtID	EnvID	Iso	QOpt	Blk
0x034B08F0	144	1	1	CS	5	B
0x034B6B10	170	1	1	CS	5	B

Dynamic SQL Variations:

Address	AnchID	StmtID	EnvID	VarID	NumRef	Typ	Lockname
0x034B1230	144	1	1	1	1	6	01000000010000000100900056
0x034B7450	170	1	1	1	1	6	01000000010000000100aa0056

Performance tuning and monitoring workshop



DB2 Performance Tuning and Monitoring Workshop (CF41)

- Audience: database designers, database administrators, application developers
- Objectives:
 - Define the impact of database design on database performance
 - Describe database application programming considerations and how they affect performance
 - Identify and describe the parameters that affect performance
 - Tune these parameters to achieve optimal performance
 - Identify and use the tools that assist in monitoring and tuning of a database

© Copyright IBM Corporation 2007

Figure 9-33. Performance tuning and monitoring workshop

CF238.3

Notes:

Overview: Learn how to tune for optimum performance the IBM DB2 UDB for Linux, UNIX, and Windows relational database management system and associated applications written for this environment. Learn about DB2 UDB for Linux, UNIX, and Windows on a serial processor, in a nonpartitioned database environment. Explore performance issues affecting the design of the database and applications using the database, the major database performance parameters, and the different tools that assist in performance monitoring and tuning.

Objectives:

- Define the impact of database design (tables, indexes, and data placement) on database performance
- Describe database application programming considerations and how they affect performance
- Identify and describe the parameters (database, non-database) that affect performance
- Tune parameters to achieve optimum performance
- Identify and use the tools that assist in monitoring and tuning of a database

Checkpoint

Exercise — Unit Checkpoint

- ___ 1. What is the database manager configuration parameter that defines the level of error logging done to the administration notification log?

- ___ 2. What tools can be used to monitor DB2 activity?

- ___ 3. What type of tool is used to determine the access strategy DB2 will use to execute an SQL statement?

Unit summary

Having completed this unit, you should be able to:

- Collect information for problem analysis and resolution
- Use error logs for basic problem analysis
- Describe four types of monitors:
 - Snapshot Monitor
 - Event Monitor
 - Activity Monitor
 - Health Monitor
- Describe the function of EXPLAIN and use this facility to assist basic analysis
- Use a series of basic commands to better work with connections and sessions
- Retrieve statistics and other information from a running DB2 instance

© Copyright IBM Corporation 2007

Figure 9-34. Unit summary

CF238.3

Notes:

Unit 10. Application issues and performance

What this unit is about

DB2 provides many options for application development. Such options include stored procedures, static and dynamic embedded SQL, call level interface (CLI), and application programming interface (API) calls.

The developer must choose the correct interface to meet the application and installation requirements. No single interface is necessarily the correct choice for all requirements that an installation may have.

Application development projects that meet installation requirements from performance, security, and business process perspectives must require application programming and administration expertise.

This unit describes the interfaces available and provides some insight regarding the features and functions of each interface. The intent of the instruction is not to provide the detailed coding requirements of each interface, but to define the interfaces at a conceptual level.

This unit also provides the database administrator with the information necessary to properly manage the database with respect to application programs.

What you should be able to do

After completing this unit, you should be able to:

- Prepare applications that access DB2 data for execution
- Use PRECOMPILE and BIND options that are appropriate for specific application requirements
- Describe the application alternatives available to access DB2 data or request other DB2 functions
- List reasons for REBIND of application programs and execute this command
- Define the concept of single dimensional clustering, and determine the proper cluster sequence, if any, for DB2 tables
- Use RUNSTATS, REORGCHK, and REORG to enhance application performance
- Work with the Explain facility

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

Administration Guide

Command Reference

SQL Reference

Unit objectives

After completing this unit, you should be able to:

- Prepare applications that access DB2 data for execution
- Use PRECOMPILE and BIND options that are appropriate for specific application requirements
- Describe the application alternatives available to access DB2 data or request other DB2 functions
- List reasons for REBIND of application programs and execute this command
- Define the concept of single dimensional clustering, and determine the proper cluster sequence, if any, for DB2 tables
- Use RUNSTATS, REORGCHK, and REORG to enhance application performance

© Copyright IBM Corporation 2007

Figure 10-1. Unit objectives

CF238.3

Notes:

10.1 Application alternatives

Application alternatives



- Database Engine Access
 - Static SQL
 - Dynamic SQL
 - Application Programming Interfaces (APIs)
- Programming Interfaces
 - Embedded SQL
 - Call Level Interface (CLI)
 - Open Database Connectivity (ODBC)
 - Java
 - Java Database Connect (JDBC)

© Copyright IBM Corporation 2007

Figure 10-2. Application alternatives

CF238.3

Notes:

As a database administrator, you will generally be involved in assisting application programmers with decisions regarding the interfaces to DB2 data. This topic does not provide details on the coding intricacies of each alternative, but will provide insight concerning the strengths and weaknesses of each alternative.

Access to the database engine from an application program is through one of three methods.

1. Static or Dynamic SQL
2. Application Programming Interfaces (APIs)

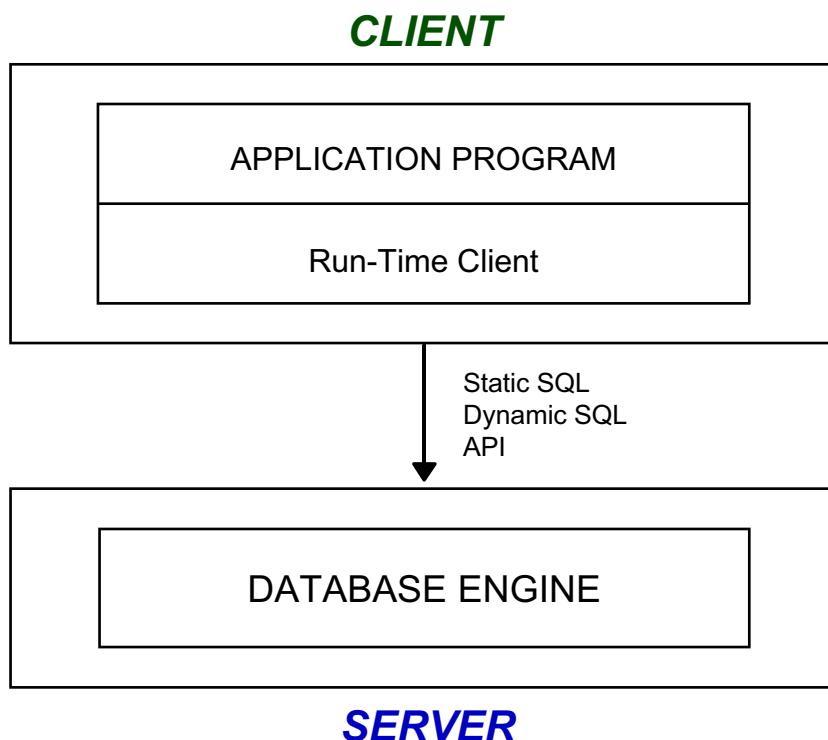
Application programs interface to the Client Application Enabler component using one of the following interfaces:

- Embedded SQL
- Call Level Interface (CLI)
 - Open Database Connectivity (ODBC)
- Java

There are also several DB2 performance features that can be implemented, such as Stored Procedures.

10.2 Database engine access

Database engine access

© Copyright IBM Corporation 2007

Figure 10-3. Database engine access

CF238.3

Notes:

An application program gets access to data through the Run-Time Client (RTC) code. An application will either:

- Request a static SQL statement to be executed on the server
- Pass a dynamic SQL statement to the server to be optimized and executed on the server
- Use a program API to request some database function to be performed on the server (such as backup, restore, load, and so forth)

The other possibility is the DB2 Run-Time Client Lite (DB2 RTCL) which is an installable component that makes it much easier to provide access to DB2 servers from Windows-based applications. The DB2 RTCL is designed to be redistributable by independent software vendors and to be used for application distribution in mass deployment scenarios. Similar to the DB2 Run-Time Client component, the DB2 RTCL provides application interfaces and the network libraries that are required for applications to run.

The main distinguishing features of the DB2 RTCL are:

- Significantly smaller disk footprint
- Shipped as a single executable, making it easy to redistribute
- Windows Installer Merge Module (.msm file) is available which simplifies the integration within a larger application

Performance — dynamic versus static SQL

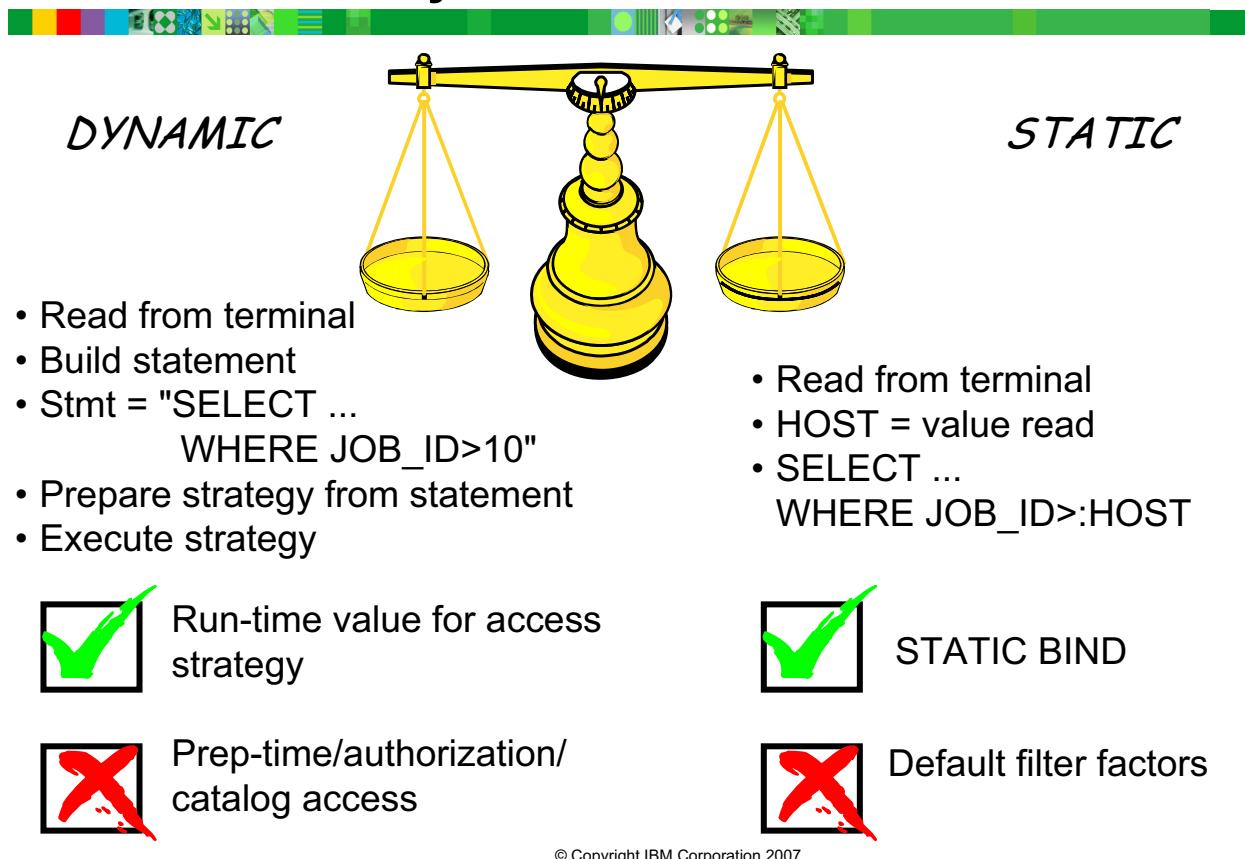


Figure 10-4. Performance — dynamic versus static SQL

CF238.3

Notes:

- Static SQL statements are statements that are fully defined at program creation time.
- SQL statements are placed in an application and are not specific to the host language.
- Variables can be used to receive data from the database manager or to provide data to the database manager. For example, assume that the fields in the sample screen are filled out by a travel agent. The application could use the Charge Card number provided to check the credit of the customer by specifying a variable on the WHERE clause of a SELECT statement. The returned credit limit could be stored in another program variable. If the credit check is passed, the application could then use the remaining information on the input screen to INSERT a row in a booking table.
- Information about the variables is present when the SQL is prepared, but the actual value of the variable is specified at run time.
- There are two major advantages of static SQL. First, preparation of the SQL is completed before execution and is not repeated every time the application runs. Second, the authorization of the package binder is used for the underlying SQL.

statements, enabling security to be based on execute authority for the program, and not on authorization to underlying objects.

- A static statement that references host variables is optimized without knowledge of the actual value of the host variable. Default filter factors are used. At times, the default filter may be too optimistic. Range predicates are sensitive to this problem. For example, consider an application with the predicate LASTNAME BETWEEN *:host1* AND *:host2*. Assume that an index on LASTNAME is chosen to support this predicate, based on default filtering. If *:host1* is set to *Abell* and *:host2* is set to *Brown* during execution, using the index would be appropriate. However, if *:host1* is set to *Abell* and *:host2* is set to *Williams* during execution, using the index would be less efficient than a table scan.
- Dynamic SQL is particularly useful in certain situations.
- Dynamic SQL is bound and optimized during execution, and all values are considered during the selection of access strategy. More accurate filter factors can be used. An access strategy that is specific to the current execution of the statement can be selected.
- Dynamic SQL, however, implies that the access strategy must be developed during execution. In many cases, even with the use of default filter factors, the total execution time of the static statements will be faster than the total execution time of the dynamic statements.

Database Manager APIs

- Application Programming Interfaces (APIs)
 - Command functions that can be included in applications
 - Enable programmatic control of various database operations
 - Used for management and maintenance of the database
 - NOT an alternative for SQL operations
- Examples
 - Start/Stop Database Manager
 - Create Database
 - Backup Database
 - Export/Import
 - Run Statistics
 - Bind

© Copyright IBM Corporation 2007

Figure 10-5. Database Manager APIs

CF238.3

Notes:

- Database manager APIs provide access to database manager functions from compiled and interpreted programming languages. These APIs are subdivided as follows:
 - Database Manager Control - Quiescing databases and instances, starting and stopping instances
 - Database Control - Creating and dropping, activating and restarting databases
 - Database Manager and Database Configuration - Getting and setting parameters
 - Database Directory Management - Cataloging and uncataloging database directory
 - Client/Server Directory Management - Cataloging and uncataloging node directory
 - Network Support - LDAP support
 - Recovery - Backup, restore, and roll forward database, manipulate history file
 - Operational Utilities - Force application, reorganization, runstats
 - Database Monitoring - Snapshot monitoring
 - Health Monitoring - Contact, notification list, contact group, alert configuration
 - Data Utilities - Export, import, load, load query
 - General Application Programming - Autoconfigure, get error or SQLSTATE messages, signal handler

- Application Preparation - Precompile, bind, rebind programs
- Remote Server Utilities - Attach, change password, detach
- Table Space Management - Table space and container query
- Partition Management - Add and drop partitions
- Satellite - Query and update satellite synchronization
- Database Partition Group Management - Redistribute database partition group
- Additional APIs - Get authorizations, instance, query and set client, read log, row and table partitioning information, archive log, inspect database

Sample program to create database with user-defined collating sequence:

```
*****credb.sqc*****
*/
/* Sample C program for "DB2 WORKSTATION INTERMEDIATE PROGRAMMING" */
/*
*/
/*
******/
/* This program will create a database. */
/*
******/

/*
*/
/* Include C library definitions */
/*
******/
#include <memory.h>      /* for memcpy */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sql.h>
#include <sqlutil.h>
#include <sqlenv.h>

/*
*/
/* Look up what the following include file contains. */
/*
******/
#include <sqle850b.h>

struct sqlca sqlca;

struct sqledbdesc dbdesc;
struct sqledbcountryinfo dbcinfo;

void sql_err();

/*
*/
/* Main routine */
/*
*/
int main()
{
int rc;
char database[9];

setbuf(stdout,NULL); /* disable output buffering */

```

```
strcpy(database, "pdb");

/* The eye-catcher *must* be initialized for the call to work.          */
memcpy(dbdesc.sqldbdid, "SQLDBD02", 8);

dbdesc.sqldbcss = SQL_CS_USER; /* this application will define the      */
/* collate sequence           */

*****?????????????????????????????????????????????????????????????*****/
/* note - memcpy is required here - sqle_850_037 is 256 bytes long,      */
/* and we are putting in it into dbdesc.sqldbudc which is 256 bytes      */
/* long - if you use a strcpy, the last byte is lost, and your new       */
/* user-defined sort sequence appears to be ignored.                      */
/*
*****                                                       */
memcpy(dbdesc.sqldbudc, sqle_850_037, sizeof(dbdesc.sqldbudc));

dbdesc.sqltsext = 2;
dbdesc.sqlcatts = (struct SQLETSDESC *) 0;
dbdesc.sqlusrts = (struct SQLETSDESC *) 0;
dbdesc.sqltmps = (struct SQLETSDESC *) 0;

strcpy(dbcinfo.sqldbcodeset, "IBM-850"); /* database codeset   */
strcpy(dbcinfo.sqldblocale, "En_US");    /* database territory */

rc = sqlecrea(database,
               (char*) 0,
               (char*) 0,

               (char) 0,
               (void*) 0,

               if (rc != 0 || sqlca.sqlcode != 0) sql_err();
               else printf("database was created successfully\n");
}

*****                                                       */
/* SQL error handler                                         */
*****                                                       */

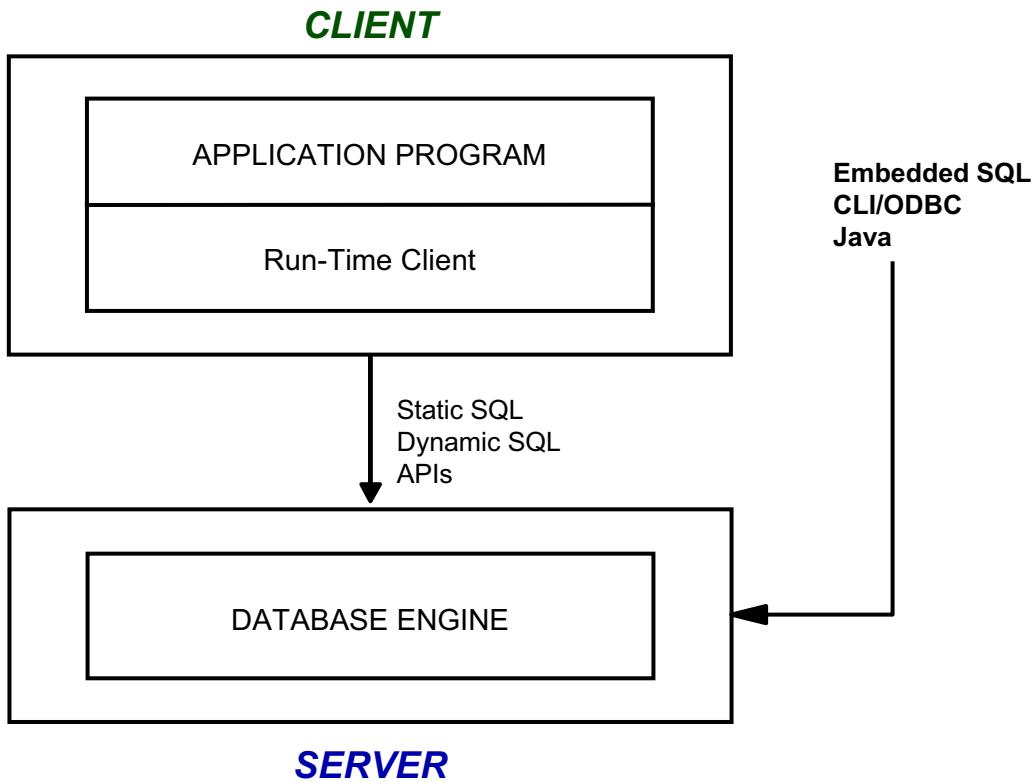
void sql_err() {
#include <sqlenv.h>
#define DATA_LEN 80
char error_message_buffer[512];
short buffer_size = sizeof(error_message_buffer);
short int rc;
int i;

printf("%s %i\n", "SQL ERROR, SQLCODE is ", sqlca.sqlcode);
/* Format the sqlca  */
rc = sqlaintp(error_message_buffer, buffer_size, DATA_LEN,
if (rc < 0)
    printf("NEGATIVE RETURN CODE FROM SQLAINTP %hi\n", rc);
else
    printf(error_message_buffer); /* Print formatted   */
```

```
    return;  
} /* end of sql_err */
```


10.3 Embedded SQL program preparation

Programming interfaces

© Copyright IBM Corporation 2007

Figure 10-6. Programming interfaces

CF238.3

Notes:

Application programs can issue SQL via the Run-Time Client component using one of several possible programming interfaces.

- Embedded SQL
- CLI
 - ODBC
- Java
 - JDBC
 - SQLJ

Embedded SQL applications

▪ Embedded SQL Applications

- SQL statements embedded in the programming language
- Can embed both Static and Dynamic SQL
- Application must be precompiled
- Need to bind the SQL and create a package at the database

© Copyright IBM Corporation 2007

Figure 10-7. Embedded SQL applications

CF238.3

Notes:

Structured Query Language (SQL) is the database interface language used to access and manipulate data in DB2 databases. You can embed SQL statements in your applications, enabling them to perform any task supported by SQL, such as retrieving or storing data. Using DB2, you can code your embedded SQL applications in the C/C++, COBOL, FORTRAN, Java (SQLJ), and REXX programming languages.



Note

The REXX and Fortran programming languages have not been enhanced since Version 5 of DB2.

An application in which you embed SQL statements is called a host program. The programming language you use to create a host program is called a host language. The program and language are defined this way because they host or accommodate SQL statements. For static SQL statements, you know before compile time the SQL statement

type and the table and column names. The only unknowns are specific data values the statement is searching for or updating. You can represent those values in host language variables. You precompile, bind, and then compile static SQL statements before you run your application. Static SQL is best run on databases whose statistics do not change a great deal. Otherwise, the statements will soon get out of date.

In contrast, dynamic SQL statements are those that your application builds and executes at run time. An interactive application that prompts the user for key parts of an SQL statement, such as the names of the tables and columns to be searched, is a good example of dynamic SQL. The application builds the SQL statement while it is running, and then submits the statement for processing.

You can write applications that have static SQL statements, dynamic SQL statements, or a mix of both.

Generally, static SQL statements are well-suited for high-performance applications with predefined transactions. A reservation system is a good example of such an application.

Generally, dynamic SQL statements are well-suited for applications that run against a rapidly changing database where transactions need to be specified at run time. An interactive query interface is a good example of such an application.

Write the program



- myapp.sqc

```

EXEC SQL DECLARE c1 CURSOR FOR
  SELECT empno,lastname FROM xyzco.templ
  WHERE wrkdept = :dept;
strcpy(dept, 'D21');
EXEC SQL CONNECT TO musicdb;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :hempno,:hlastname;
while (sqlca.sqlcode==0){

  EXEC SQL FETCH c1 INTO :hempno,:hlastname;
} /*end while */

EXEC SQL CLOSE c1;
EXEC SQL CONNECT RESET;

```

© Copyright IBM Corporation 2007

Figure 10-8. Write the program

CF238.3

Notes:

SQL statements placed in an application are not specific to the host language. The database manager provides a way to convert the SQL syntax for processing by the host language:

- For the C, C++, COBOL, or FORTRAN languages, this conversion is handled by the DB2 precompiler. The DB2 precompiler is invoked using the PREP command. The precompiler converts embedded SQL statements directly into DB2 run-time services API calls.
- For the Java language, the SQLJ translator converts SQLJ clauses into JDBC statements. The SQLJ translator is invoked with the sqlj command.

When the precompiler processes a source file, it specifically looks for SQL statements and avoids the non-SQL host language. It can find SQL statements because they are surrounded by special delimiters.

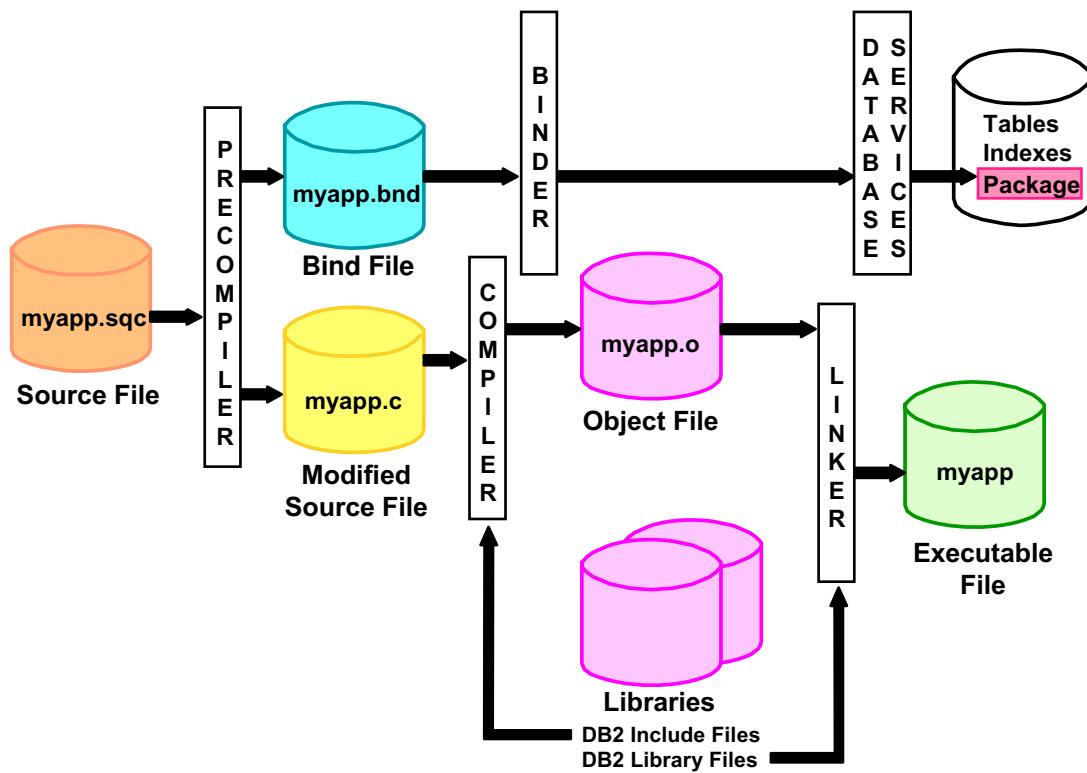
The extension of the source module identifies the source language in the embedded SQL program. Supported extensions include:

- **.sqc** for C files on all supported platforms
- **.sqC** for C++ files on UNIX platforms
- **.sqx** for C++ files on Windows operating systems
- **.sqb** for COBOL files on all supported platforms, though the TARGET precompile option allows for any file extension
- **.sqf** for FORTRAN files on all supported platforms, though the TARGET precompile option allows for any file extension
- **.sqlj** for SQLJ files on all supported platforms

The program on the graphic is provided for illustrative purposes. A brief description of its function:

- First, the program **DECLares** the cursor, stating what **SELECT** statement it will be used with, and whether updates will be performed using it. Then, the program **OPENS** the cursor. This allows the application to gain access to the data that meets or will meet the definition of the cursor. (If host variables were in the definition of the cursor, the value of the host variables at OPEN would be used in the statement.)
- The program uses **FETCH** statements to move the cursor forward and to look at each row in the result set. When the program has finished processing the results of the query, it **CLOSES** the cursor, which destroys the result set.

Program preparation steps



© Copyright IBM Corporation 2007

Figure 10-9. Program preparation steps

CF238.3

Notes:

When you embed SQL statements in your application, you must precompile and bind your application to a database with the following steps:

1. Create source files that contain programs with embedded SQL statements.
2. Connect to a database, then precompile each source file.

The precompiler converts the SQL statements in each source file into DB2 run-time API calls to the database manager. The precompiler also produces an access package in the database and, optionally, a bind file, if you specify that you want one created. The access package contains access plans selected by the DB2 optimizer for the static SQL statements in your application. The access plans contain the information required by the database manager to execute the static SQL statements in the most efficient manner as determined by the optimizer. For dynamic SQL statements, the optimizer creates access plans when you run your application.

The bind file contains the SQL statements and other data required to create an access package. You can use the bind file to rebind your application later without having to precompile it first. The rebinding creates access plans that are optimized for current database conditions.

You need to rebind your application if it will access a different database from the one against which it was precompiled. You should rebind your application if the database statistics have changed since the last binding.

3. Compile the modified source files (and other files without SQL statements) using the host language compiler.
4. Link the object files with the DB2 and host language libraries to produce an executable program.
5. Bind the bind file to create the access package if this was not already done at precompile time, or if a different database is going to be accessed.
6. Run the application. The application accesses the database using the access plan in the package.

The above visual represents the program preparation process. The following notes summarize the activities that occur during each step.

1. PRECOMPILE
 - The correct precompiler is determined based on the extension of the source code.
 - SQL statements are syntactically checked. Then, the SQL statements are commented out in the source code and replaced with host language calls. The SQL in the source file is found by the precompiler through the use of language specific delimiters. The resultant output is called the **modified source file**.
 - Either a **bind file** or a **package** is also created. (It is also possible to create both.) The default is to create a package. However, there are several reasons to override the default and create a bind file. For discussion purposes, assume that the example shown uses a precompiler option that creates a bind file.
2. COMPILE
 - The modified source file can then be processed using the compiler appropriate for the chosen language. There is no SQL in the modified file so that it can be processed.
 - Compile will resolve INCLUDEs commonly used by application programmers. In addition to those libraries *normally* used by a particular installation, a DB2 library containing INCLUDEs relevant to the DB2 programming interface needs to be identified. This identification can be done through compiler command options if symbolic links (AIX), updates to system environment variables (Windows NT), or some other site-specific technique has not been used to place the location of the INCLUDE files into the path used by default to find INCLUDE files. Consult the product documentation for further details.
3. LINK
 - Assuming that there are no errors discovered during compile, the resultant object module can be linked.
 - The linker resolves references to run-time services DB2 APIs that were placed in the modified source file by the precompiler, in addition to all other module references needed by the *non-SQL* code. The library needed for the DB2 calls can be explicitly included as a command option on LINK if it has not been identified through techniques similar to those listed for the compile step.

4. BIND

- The *SQL component* of the application that is present in the bind file is analyzed.
- Syntax and authorization are checked against the target database.
- Assuming that there are no syntax or authorization problems, an access strategy is determined for each SQL statement. This strategy is stored in a DB2 object called a **package**. More details concerning the generation of access strategies, called **optimization**, is presented in the Application Performance unit.
- DYNAMICRULES - There is a requirement to be able to lend the DML privileges of a package binder to the package executor during the execution of dynamic SQL. Some businesses have, or are heading towards having, applications with large amounts of dynamic SQL, coupled with the need to allow a large and potentially unknown user base access to the information the SQL is allowed to access. Web-initiated query tools are a good example. A user is given the opportunity to receive information based on a query generated from criteria they have supplied. SQL gets dynamically generated and the information is retrieved. When the application is bound with DYNAMICRULES BIND, access to the tables that contain the query information only have to be given to the package binder and not to every user who might need access to the information; the authorization ID of the package owner is used for authorization checking, not that of the user. An application bound with DYNAMICRULES RUN, the default, requires that the user has access to the tables that contain the query information. References to unqualified objects will be the CURRENT SCHEMA of the user.

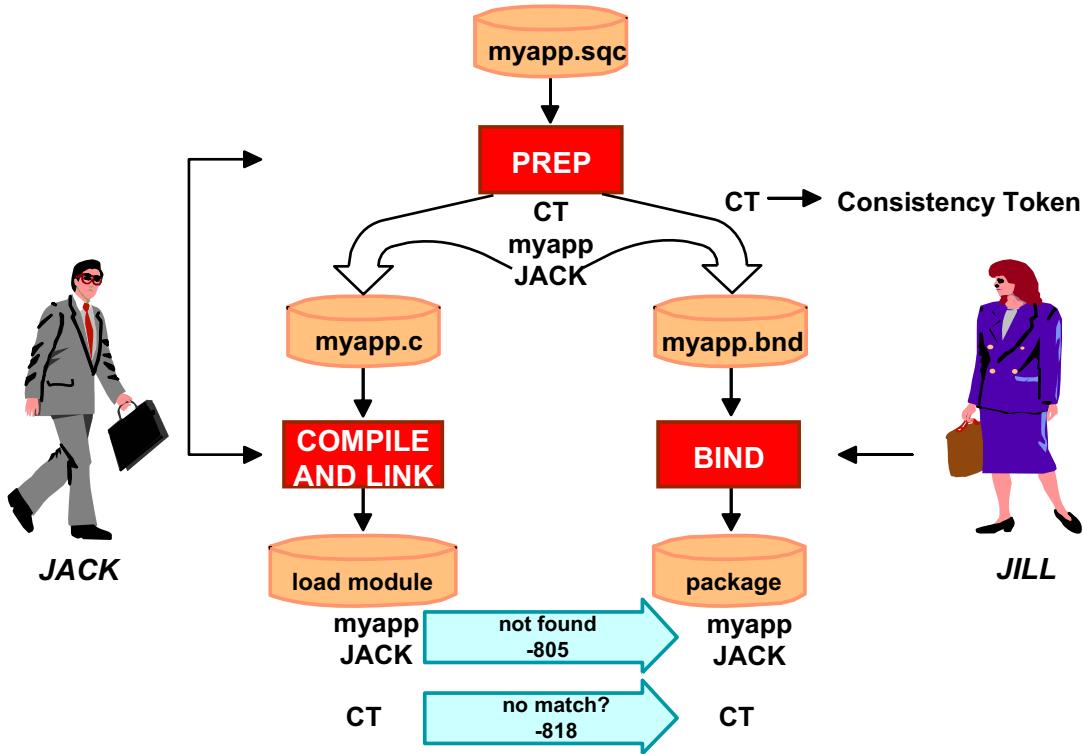
Reasons to separate the BIND step from the PRECOMPILE step include:

1. The compile and link steps can be successfully completed before using database resources. Many programming errors can be detected and corrected before using database manager resources.
2. The package must be bound against more than one database.
3. Authorization of the ID used to do the BIND is different from that of the ID used to do the precompile.

During execution, the executable will pass the full package name, statement section number, and consistency token to the database manager. This allows DB2 to locate and execute the correct access strategy for the executable. This is also the architecture that supports executables that contain more than one object module, since each executable SQL statement results in an independent call to DB2 with specific information concerning the package and statement.

In order to execute a section of a package, the section must be loaded into memory. The size of the memory area allocated at the database level to support package sections is designated via the configuration parameter **pckcachesz**. If sections of SQL that may be executed on a repetitive basis can be kept in the cache, a performance benefit is realized because the database manager does not need to reload the section. This could be of significant benefit to transactional systems.

Associating load modules and packages



© Copyright IBM Corporation 2007

Figure 10-10. Associating load modules and packages

CF238.3

Notes:

During PRECOMPILE, a consistency token is placed in both the modified source and the bind file. This token is necessary to ensure that the resultant load module and package were derived from the same original source. (If a bind file is not created, the token is placed directly in the resultant package.)

Also at PRECOMPILE time, the name of the package that the executable load module will request is stored in the modified source module. By default, this will be the same as the name of the source module (the name excluding .sqc), truncated to eight characters.

A consistency token contains a timestamp of when the precompile was done. It is often referred to as a *timestamp*, but we will use the term *consistency token* both to avoid ambiguity and for consistency with other platforms.

When a call to the database manager is made by an application, the package creator, package name, and consistency token are passed to the database manager.

If a package that matches the creator and name cannot be found, the application receives a -805 SQLCODE.

If a package is found, the consistency token passed by the load module is compared to the token in the package. If the values match, this indicates that the load module and package were created from the same original source and execution proceeds. If the tokens do not match, the application receives a -818 SQLCODE which indicates that the load module and package were created from sources created by different precompiles.

Any changes to original source code require both SQL and non-SQL program preparation steps to be completed.

Why might you want to separate the precompile and bind activities and have them done by different individuals?

Answer: If you request a bind file at precompile time, certain object existence and authentication SQLCODES are treated as warnings instead of errors. This enables you to precompile a program and create a bind file without requiring that the referenced objects be present, or having to be authenticated. So, Jack would not need to have the authority to access the objects referenced in the SQL, as long as Jill did when she did the bind.

Precompiling versus binding

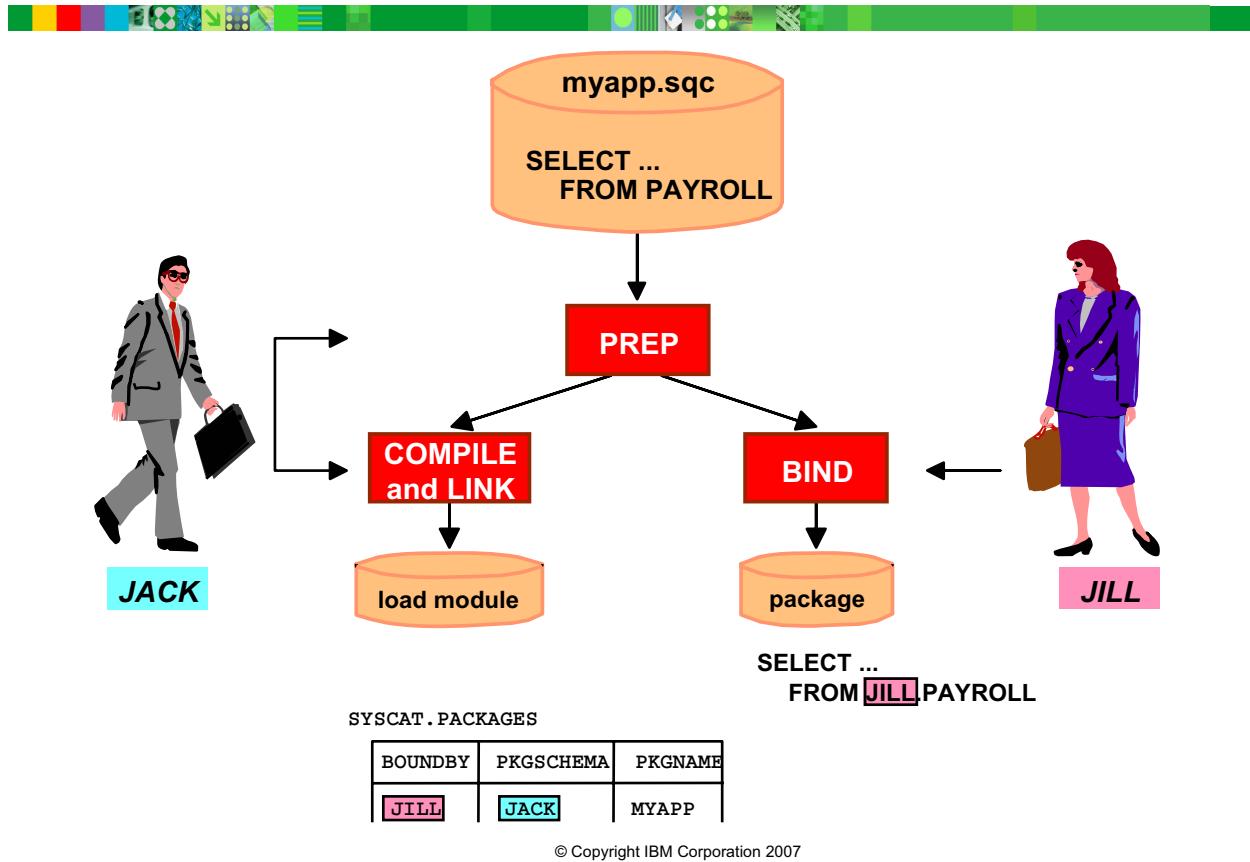


Figure 10-11. Precompiling versus binding

CF238.3

Notes:

Unqualified references in applications are qualified with the ID of the owner in the package definition.

The value in the `BOUNDBY` column in the `SYSCAT.PACKAGES` table specifies the “owner” of the package being bound. That will be the binder’s ID only if an “owner” parameter is not specified on the bind.

It is possible that JACK has access to a test table named `JACK.PAYROLL`. When JACK precompiles the code, syntax and authorization will be done against his test table. When JILL does the production BIND, the target table will be `JILL.PAYROLL`.

If the application contained fully-qualified references, for example:

```
SELECT ....
  FROM PROD3.PAYROLL
```

then the binder would need authorization against the `PROD3.PAYROLL` object, but the ID of the binder would not be used in qualification, since the reference is already fully defined.

Adjusting qualifier and owner



Jill issues:

```
db2 bind myapp.bnd  qualifier u1  owner u2
dynamicrules bind
```

- Unqualified SQL uses **u1** schema
- **u2** owns package
 - Can drop, rebind, grant privileges
 - **u2's** privilege checked for dynamic SQL

© Copyright IBM Corporation 2007

Figure 10-12. Adjusting qualifier and owner

CF238.3

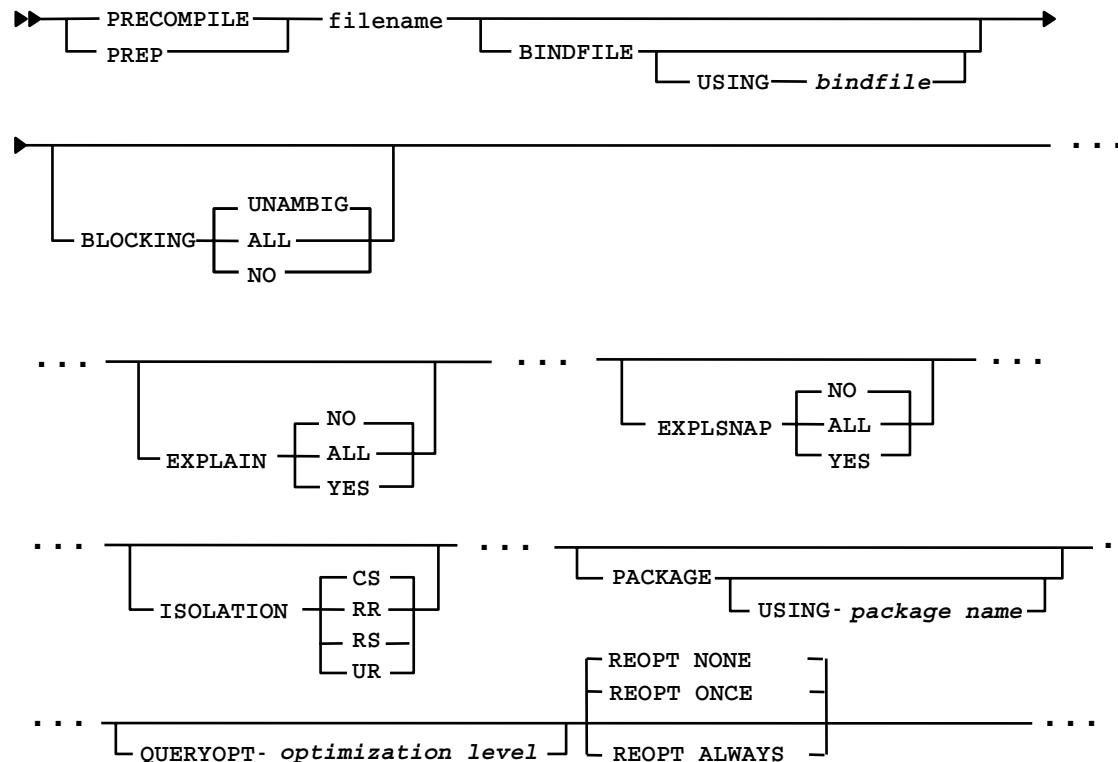
Notes:

QUALIFIER provides an 8-character implicit qualifier for unqualified objects contained in the package. The default is the owner's authorization ID, whether or not OWNER is explicitly stated.

OWNER designates an authorization identifier for the package owner. The owner must have the privileges required to execute the SQL statements in the package. The default is the primary authorization ID for the precompile/bind process if this option has not been explicitly specified.

The DYNAMICRULES option and its values can be supplied at BIND time, and define which rules apply to dynamic SQL at run time for the authorization id, and for the initial setting of the value used for implicit qualification of unqualified object references. Valid DYNAMICRULES options are RUN and BIND; RUN is the default.

Precompile command syntax (basic)



© Copyright IBM Corporation 2007

Figure 10-13. Precompile command syntax (basic)

CF238.3

Notes:

PRECOMPILE or PREP can be used interchangeably.

A database connection must exist before this command is executed. The syntax allows many options to be specified, but not all would be required in every case. As a simple example, consider the following.

```
db2 connect to musicdb
db2 prep myapp.sqc bindfile
db2 connect reset
```

This would run the precompiler and create the bind file. It would not create a package.

A package will be created if you:

1. Use both the BINDFILE option and the PACKAGE option
2. Use the BINDFILE option and then the DB2 bind command
3. Omit the BINDFILE option

The capability to explicitly name the bind file, package, or both is provided through the USING option. However, it is not recommended, since the default names can easily be related to the original source. For example, assume that you are preparing myapp.sqc; the default bind file would be myapp.bnd and the default package would be myapp.

BLOCKING allows the data transfer from server to client for READ-ONLY cursors to be in multiple row blocks instead of on a row-by-row basis. Greater detail regarding the meaning of the above settings for BLOCKING will be provided later in this topic.

EXPLAIN stores information in the EXPLAIN tables about the access plans chosen for each SQL statement in the package, if the YES option is chosen on the EXPLAIN parameter. The ALL value will collect information on each eligible static SQL statement. Information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to NO. DRDA does not support the ALL value for this option. The NO option on the EXPLAIN parameter means that EXPLAIN information will not be collected. With the REOPT option, EXPLAIN information for each reoptimizable incremental bind SQL statement will be placed in the EXPLAIN tables at run time. In addition, Explain Snapshot information will be gathered for reoptimizable dynamic SQL statements at run time.

EXPLSNAP determines if an EXPLAIN snapshot should be stored in the Explain Tables. ALL specifies that an EXPLAIN snapshot will be created for static and dynamic SQL statements when the statement is optimized. YES specifies that only static statements should be EXPLAINed. NO disables creating an EXPLAIN snapshot. With the REOPT option, EXPLAIN information for each reoptimizable incremental bind SQL statement will be placed in the EXPLAIN tables at run time. In addition, EXPLAIN snapshot information will be gathered for reoptimizable dynamic SQL statements at run time.



Note

EXPLAIN captures data for use with the db2exfmt tool, while EXPLSNAP captures data for use with the Visual Explain tool.

ISOLATION determines the degree to which this application will be isolated from the changes made by other applications. This option impacts the acquisition and duration of row share locks. The options are abbreviations for Cursor Stability (CS), Repeatable Read (RR), Read Stability (RS), and Uncommitted Read (UR).

QUERYOPT determines the level of optimization that should be performed. The higher the value (up to a maximum of 9), the greater the number of access paths considered.

REOPT specifies whether to have DB2 optimize an access path using values for host variables, parameter markers, and special registers. Valid values are:

- **NONE:** The access path for a given SQL statement will not be optimized using real values for the variables. The default estimates for these variables will be used instead, and this plan is cached and used subsequently. This is the default.

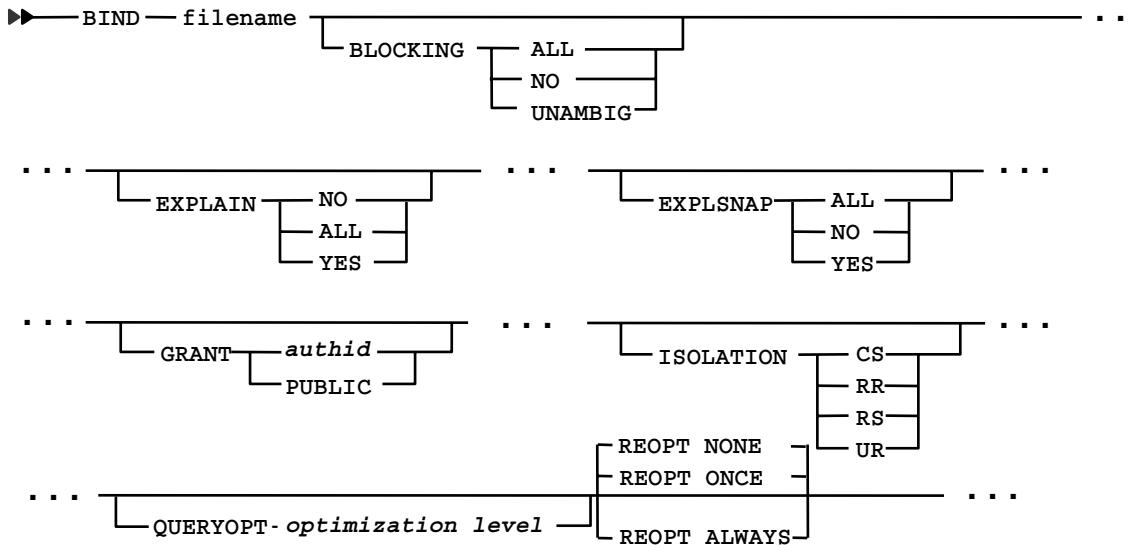
- **ONCE:** The access path for a given SQL statement will be optimized using the real values of the host variables when the query is first executed. This plan is cached and used subsequently.
- **ALWAYS:** The access path for a given SQL statement will always be compiled and reoptimized using the values of the host variables known at each execution time.

The PRECOMPILE command in a DRDA environment has options specific to that environment. Also, other options for the PRECOMPILE command exist. Consult the product documentation if desired.

References

Command Reference

Bind command syntax (basic)



© Copyright IBM Corporation 2007

Figure 10-14. Bind command syntax (basic)

CF238.3

Notes:

A database connection must exist before this command is executed. The syntax allows many options to be specified, but not all would be required in every case. As a simple example, consider the following.

```
db2 connect to musicdb
db2 bind myapp.bnd blocking all
db2 connect reset
```

This would BIND the statements in the specified bind file and use the option to block all cursors that are not explicitly updatable.

The filename specified on this command must have an extension of bnd.

It is possible to bind multiple packages by specifying a filename that is actually a list of bind files. The syntax for specifying a list is to preface the name of the file with the **at sign (@)**. The bind files in the list file are designated according to the following syntax:

```
first.bnd+second.bnd+
third.bnd+ (etc) n.bnd+
last.bnd
```

The first position in any line cannot contain the + sign, and the last line should not contain the + sign.

Options specified on the BIND command will take precedence over any options specified or defaulted on the PRECOMPILE command. Options not specified will use the value specified or defaulted at precompile.

The GRANT option of the BIND command is one that is not available on the PRECOMPILE command. If it is specified, BIND and EXECUTE authority are granted to PUBLIC or the authid named.

The BIND command in a DRDA environment has options specific to that environment. Also, other options for the BIND command exist. Consult the product documentation if desired.

With the BIND command, the several REOPT options can certainly also be used.

There is another option available with bind: **ACTION**.

ACTION indicates whether the package can be added or replaced.

- ADD indicates that the package does not exist and that a new package is to be created. If the package exists, the execution will be stopped and a error message is returned.
- REPLACE indicates that the existing package is to be replaced by a new one with the same package name and creator. This is the default for the ACTION option.
 - REPLVER *version-id* could be specified as an additional option for REPLACE. If this option is not specified and a package already exists that matches the name, creator, and version of the package being bound, then that package will be replaced; if not, then a new package will be added.

References

Command Reference

Package catalog views



■ SYSCAT.PACKAGES

- One row for each package on the system
- Information about parameters, schema, binder, and so forth

■ SYSCAT.PACKAGEAUTH

- One row for every privilege held on packages
- Information about grantor, grantee, privilege set

■ SYSCAT.PACKAGEDEP

- One row for each dependency of the package
- Information about indexes, tables, views, triggers, functions, and aliases needed by the package

© Copyright IBM Corporation 2007

Figure 10-15. Package catalog views

CF238.3

Notes:

The catalog views described above can be useful when managing the applications for an installation.

The following items are some examples of how the catalog views could be used.

- PACKAGES

- You need to determine why a READ-ONLY application is running slowly. You suspect that the blocking option was improperly chosen. The column BLOCKING provides you with the information.
- A package that accesses multiple servers is periodically returning -818 SQLCODES. You know that a change to the code was done on Friday night. The column EXPLICIT_BIND_TIME provides you with the time that the last explicit bind was done, so you could determine the servers that have an older version of the package (older than Friday's date.)

- PACKAGEAUTH

- You are asked by an auditor to list all IDs that can execute a particular sensitive application. The columns GRANTEE and GRANTEETYPE could be queried for the package for rows where the EXECUTEAUTH column contains a Y.
- You are about to go on vacation, and your backup needs the authority to BIND your applications while you are away. The BINDAUTH column indicates whether or not such an authority is possessed.

- PACKAGEDEP

- You are considering dropping an index that you believe to be rarely used. The columns PKGSCHEMA and PKGNAME could be queried to list those packages that are currently using the index in an access strategy.
- You wish to list all the tables and views that an application accesses. The columns BSCHEMA and BNAME could be queried for the given package for the BTYPE values of V (View) and T (table).

All the columns defined by the views are described in the *SQL Reference*.

References

SQL Reference

Application process



TASK

STATEMENT

Connect to musicdb	db2 connect to musicdb
Prepare C Source Code	db2 prep myapp.sqc bindfile
Compile and Link	icc myapp.c db2api.lib
Bind at Local	db2 bind myapp.bnd blocking all
Reset Connection	db2 connect reset
Execute Object Code	myapp > myapp.out
Edit Output	more < myapp.out

© Copyright IBM Corporation 2007

Figure 10-16. Application process

CF238.3

Notes:

The above example uses a source code program named *myapp*, which is written in C language with embedded SQL statements that access a DB2 database called *musicdb*.

SQLLIB/samples directory has sample programs for a wide variety of languages and interfaces, including .NET, C/C++, CLP, COBOL, java, perl, php, SQL, Stored Procedures, Visual Basic, Visual C, and XML.

Developer Workbench and SQL PL

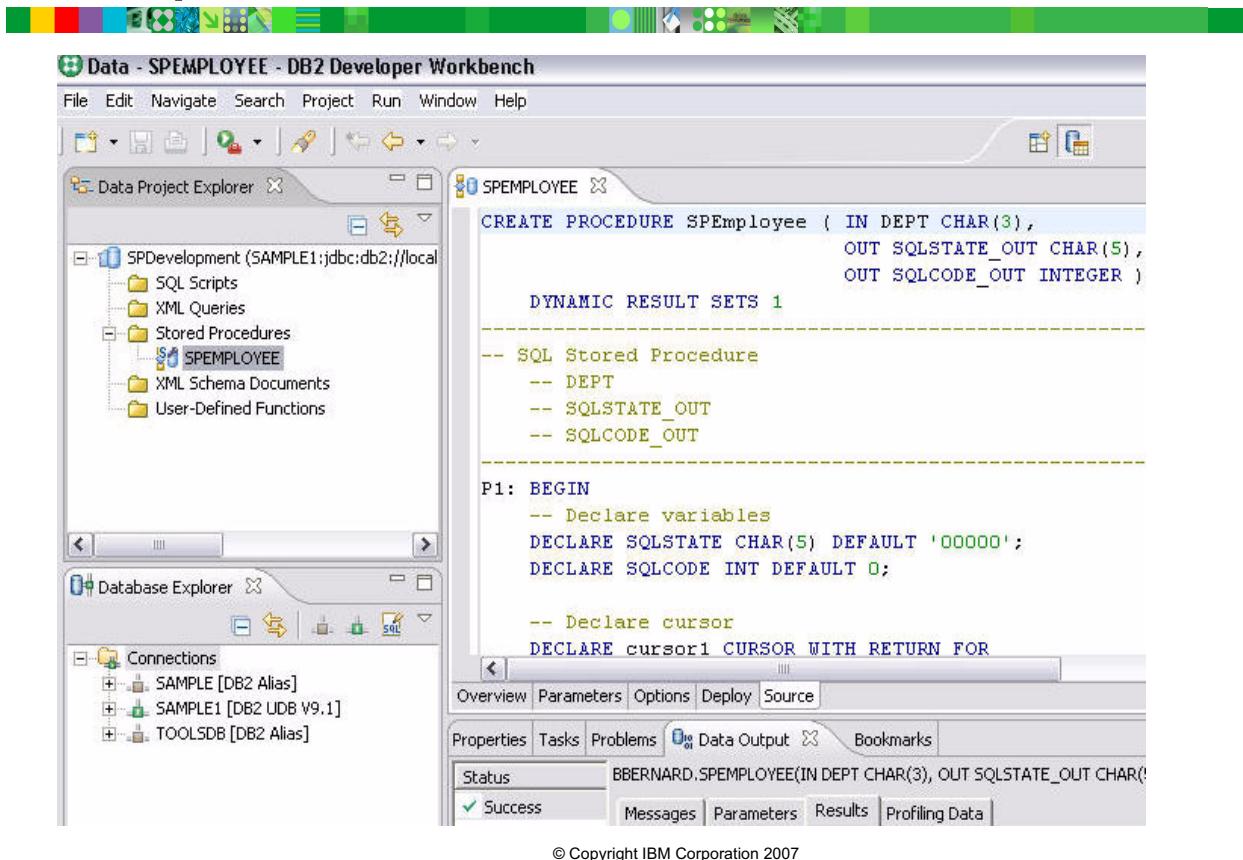


Figure 10-17. Developer Workbench and SQL PL

CF238.3

Notes:

DB2 Developer Workbench is an Eclipse-based tool that replaces the Development Center from DB2 UDB for Linux, UNIX, and Windows, Version 8. Developer Workbench is a comprehensive development environment for creating, editing, debugging, deploying, and testing DB2 stored procedures and user-defined functions. You can also use Developer Workbench to develop SQLJ applications, and to create, edit, and run SQL statements and XML queries.

The Developer Workbench includes functionality that is comparable to Development Center, including these items:

- Database code designers for SQL statements, stored procedures, and user-defined functions
- Support for running statements in the database
- Debugging support for DB2 stored procedures

In addition to existing Development Center functionality, there are some additional new features:

Migrate existing Development Center projects - You can use a wizard to migrate existing Development Center projects into Developer Workbench.

Compare routines - You can use the compare editor to compare and make changes between two routines that are contained within a data development project in Developer Workbench. You can also compare routine attributes for routines that are stored on a server.

Deploy routines to unlike servers - You can deploy routines that were created for one DB2 database to a DB2 database on a different platform. For example, you can create a routine for a DB2 Database for Linux, UNIX, and Windows and then deploy it to a DB2 UDB for z/OS database. Not all server combinations are supported.

Deploy binaries - For SQL or Java stored procedures targeting DB2 UDB for z/OS, Version 8 or higher, you can deploy without going through a full rebuild. The binaries for a SQL procedure or the JAR file for a Java procedure are copied from the source to the target system.

Launch Visual Explain - You can launch Visual Explain for DB2 UDB for z/OS or DB2 for Linux, UNIX, and Windows SQL statements from either the routine editor for SQL routines or from the wizard for routine creation.

Develop SQLJ applications - You can develop SQLJ applications by using the following features:

Generate an SQLJ template file by using a wizard - Translate and compile SQLJ files automatically.

Customize and bind SQLJ profiles by using a wizard - Print SQLJ profile files.

Edit applications by using code assist and templates - Debug SQLJ files.

Share project resources - You can share your Developer Workbench data development project by using either Concurrent versions System (CVS) or ClearCase. After you share your project, you can manage all changes and update history, and you can synchronize your files with the repository.

Edit table data - You can use an editor to edit the data that is contained in a table. You can edit existing values, delete an existing row, or insert a new row.

Extract and load data - You can extract the data from a table or view into a file on the local file system. You can use this file to load the data into a table.

Debug stored procedures - Developer Workbench includes enhanced integrated stored procedure debugging capabilities. You can debug SQL or Java stored procedures for supported DB2 servers, or Java stored procedures for supported Derby servers.

XML support - Developer Workbench contains support for XML functions, the XML data type, and XML schema registration. You can also create XQueries with the XQuery builder.

Learn about Developer Workbench through an information center and tutorials - Developer Workbench help and tutorials are available in an information center that is installed with Developer Workbench. This information is for Developer Workbench only,

and it is not installed with the DB2 information center CD. To access the Developer Workbench information center, click **Help > Help Contents** from the main menu in the product. You can also link directly to important getting started information from the Welcome page in Developer Workbench by clicking **Help > Welcome**.

10.4 Application performance

DB2 Optimizer

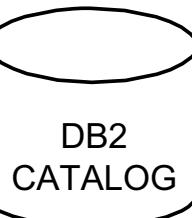


SQL STATEMENTS

Optimizer checks authorizations, determines what data is requested, and from statistics, determines an access plan

ALTERNATE COST ALGORITHMS

STATISTICS



Then selects one

PACKAGE (Access Path)

The Optimizer Chooses:

- The order in which tables are accessed
- Which index to use
- The join method
- The type of scan required

© Copyright IBM Corporation 2007

Figure 10-18. DB2 Optimizer

CF238.3

Notes:

During the bind process, the optimizer will consider several factors to determine the access strategy.

The syntax constructs of the SQL statement itself will be analyzed. For example, consider the unique index on ARTNO in the ARTISTS table. A predicate of the form `ARTNO <> value` cannot usually take advantage of the index via a matching index scan, while a predicate of the form `ARTNO = value` could.

The size of tables, the number of unique values in columns and indexes, and the second highest and second lowest values in columns are just some examples of another key input for access strategy determination: catalog statistics. RUNSTATS is the utility that gathers these statistics.

The indexes available for consideration are available to the optimizer via the catalog. As a reminder, an index may be used to access particular data values or to avoid a sort.

The sizes of the buffer pools are another key input for the bind process. The size of the buffer pool determines the number of index and data pages that can be cached in it. Large

buffer pools can reduce the number of I/O operations necessary to satisfy SQL statements. Therefore, the size of the buffer pool can influence the optimizer because I/O costs are a major part of the overall cost formulas for access strategies.

The estimate of the average number of applications, indicated by the configuration parameter **avg_appls**, can affect optimization. This parameter can be set between 1 and maxappls. If it is set to 1, the optimizer assumes that it has the entire buffer pool to use to support the statement. If it is set to higher values, the optimizer may consider strategies that burden the buffer pool to a lesser degree. In an environment that supports a large number of complex adhoc query requests, specifying a higher value may be appropriate to reflect the demand that will exist on the buffer pool.

The CPU speed of the machine, which can be indicated via the database configuration parameter **cpuspeed**, is also used by the optimizer as part of its cost evaluation. Normally, allowing DB2 to determine this value would be sufficient. Consult the reference material for further details.

Rather than optimizing a query once, when it is compiled, the Learning Optimizer (LEO) watches production queries as they run and then fine-tunes them as it learns about data relationships and user needs. This is helpful in large and complex databases.

LEO automatically self-validates the query optimizer's cardinality model by instrumenting the execution module to collect actual cardinalities at each step of the query execution plan. After the query completes, LEO compares these actuals to the query optimizer estimates to produce adjustment factors to be exploited by future optimization of queries that have similar predicates.

The approach is very general and can correct the result of any sequence of operations, not just the access of a single table or even just the application of predicates.

Optimization classes



- QUERYOPT optimization level

- 0 Use a minimal amount of optimization
- 1 Use a degree of optimization roughly comparable to DB2/6000 Version 1, plus some additional low-cost features not found in Version 1
- 2 Use features of opt level 5, but simplified join algorithm
- 3 Perform a moderate amount of optimization; similar to the query optimization characteristics of DB2 for z/OS
- 5 Use a significant amount of optimization; with Heuristic Rules (default)
- 7 Use a significant amount of optimization; without Heuristic Rules
- 9 Use all available optimization techniques

© Copyright IBM Corporation 2007

Figure 10-19. Optimization classes

CF238.3

Notes:

When an SQL query is compiled, a number of optimization techniques can be used to determine the most efficient access plan for that query. Using more optimization techniques increases query compilation time as well as system resource usage. For this reason, you may wish to limit the number of techniques applied to optimizing your query by setting the optimization level when creating the package.

For example, if you know that the statements in an application are simple, perhaps with predicates that reference indexes with a high degree of cardinality, you may not want to expend the resources to optimize at a high degree.

A greater level of optimization may be justified for complex queries, since the cost spent at optimization may pay substantial benefit from the resulting access strategy. In other words, saving on optimization may result in an access strategy that is inefficient.

For dynamic SQL applications, the value of the **CURRENT QUERY OPTIMIZATION** special register will be used to determine the class to be used. This register can be updated using the **SET CURRENT QUERY OPTIMIZATION=n** statement, where n is a value of 0, 1, 2, 3, 5, 7, or 9, or a host variable containing one of these values.

More detail regarding optimization is provided in the *Administration Guide*.

Adjusting the QUERY OPTIMIZATION class

- Database Configuration Parameter

```
db2 UPDATE DB CFG FOR musicdb USING DFT_QUERYOPT n
```

- Command

```
db2 SET CURRENT QUERY OPTIMIZATION=n
```

- Prep or Bind option

```
db2 PREP pgml.sqc QUERYOPT n
```

- db2cli.ini file for CLI clients

```
DB2OPTIMIZATION=n
```

© Copyright IBM Corporation 2007

Figure 10-20. Adjusting the QUERY OPTIMIZATION class

CF238.3

Notes:

DFT_QUERYOPT defines the default optimization class used by the optimizer. This value can be set using the command:

```
db2 UPDATE DB CFG FOR <database> USING DFT_QUERYOPT n
```

where n is one of the optimization classes.

The optimization level used on dynamic SQL statements submitted through the CLP can be set using the command:

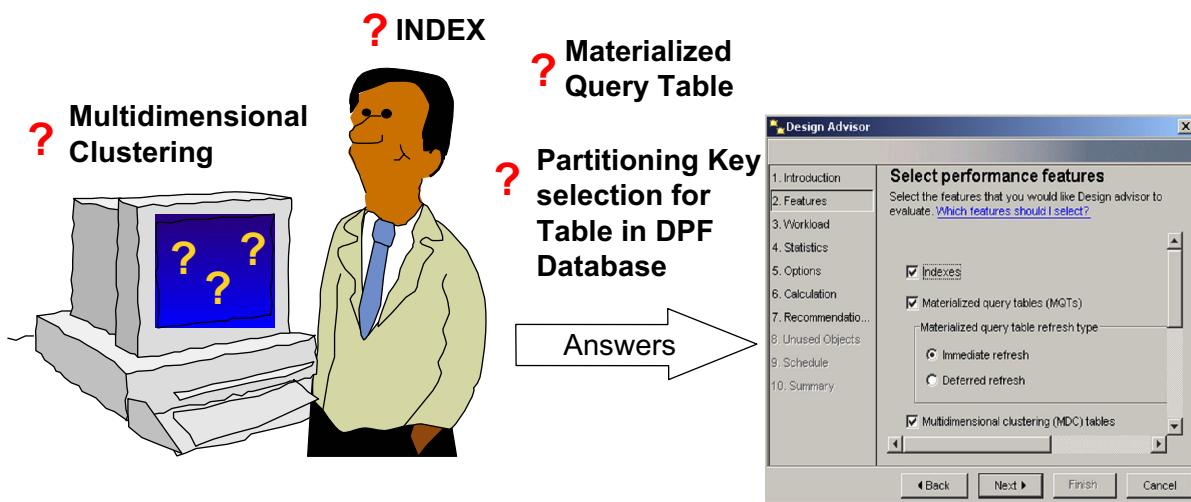
```
db2 SET CURRENT QUERY OPTIMIZATION=n
```

where n is one of the optimization classes.

- For an embedded SQL program, the optimization level can be set at PREP or BIND using the QUERYOPT n parameter.
- For dynamic SQL applications, the value of the **CURRENT QUERY OPTIMIZATION** special register can be used to determine the optimization level to use on subsequent dynamic SQL statements.
- For JDBC, ODBC or CLI applications, the value of **DB2OPTIMIZATION** set in the db2cli.ini file will set the optimization level.

Design Advisor

- Assists in finding the right indexes, MQTs or MDC
 - Based on workload
 - Virtual indexes
- Reduce complexity of performance analysis and tuning



© Copyright IBM Corporation 2007

Figure 10-21. Design Advisor

CF238.3

Notes:

Appropriate indexes have great impact on the access plans selected and the performance of SQLs.

The Design Advisor, which can be invoked by right-clicking a database and selecting Design Advisor, is a tool that can help you significantly improve your workload performance. The task of selecting which indexes, MQTs, clustering dimensions, or partitions to create for a complex workload can be daunting. The Design Advisor identifies all of the objects needed to improve the performance a particular workload (can include SELECT, INSERT, UPDATE, and/or DELETE statements). Given a set of SQL statements, it will generate recommendations for:

- New indexes
- New materialized query tables (MQTs)
- Conversion to multidimensional clustering tables (MDC)
- Repartitioning of tables
- Deletion of indexes and MQTs unused by the specified workload

You can decide to implement some or all of the recommendations immediately or schedule them for a later time.

The Design Advisor can also help you to migrate from a single-partition database to a multi-partitioned-environment.

For example, over a one month period of time your database manager may have to process 1,000 INSERTs, 10,000 UPDATEs, 10,000 SELECTs, and 1,000 DELETEs. The information in the workload is concerned with the type and frequency of the SQL statements over a given period of time. The advising engine uses this workload information in conjunction with the database information to recommend indexes. The goal of the advising engine is to minimize the total workload cost. This information is written to the ADVISE_WORKLOAD table. With sufficient information/constraints, the Design Advisor is able to suggest the appropriate actions to take for your tables.

You can certainly run the Design Advisor from the command line, then the output is printed to stdout by default and saved in the ADVISE_TABLE and ADVISE_INDEX tables.

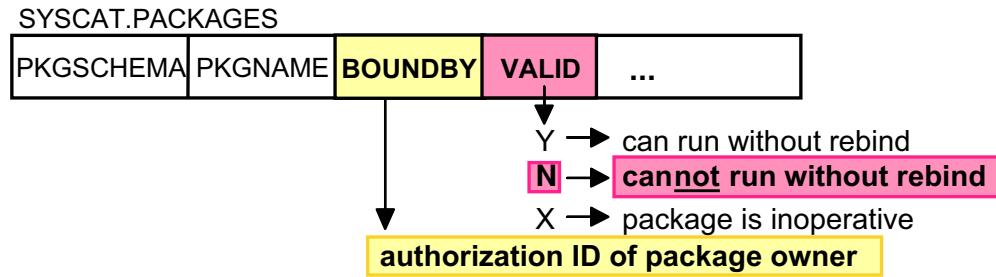
Partitioning strategies can be found in the ADVISE_PARTITION table. The RUN_ID value in all these tables corresponds to the START_TIME value in the ADVISE_INSTANCE table for each execution of the Design Advisor.

To create the ADVISE_WORKLOAD and ADVISE_INDEX tables, run the EXPLAIN.DDL script found in the misc subdirectory of the sqllib subdirectory. If not already created, the Design Advisor can also create the tables.

Implicit rebind

- **Implicit rebinding** is automatic at next execution and triggered if:

- Drop object referenced in package
- Drop primary key on table referenced in package
- Drop/add referential constraint on parent or child table referenced in package
- Revoke **privilege required by owner** to execute static SQL statement embedded in package
- Requested using the REOPT option



© Copyright IBM Corporation 2007

Figure 10-22. Implicit rebind

CF238.3

Notes:

Implicit rebinding is available on DB2. It is totally automatic at execution time and is triggered under the above circumstances. When SYSCAT.PACKAGES.VALID equals **N**, then an implicit rebind is triggered at execution time. When an implicit rebind is performed, the access paths for all the SQL statements are rebound. The BOUNDBY authorization ID, the ID of the original package owner, is checked for the appropriate authority and privileges to execute the embedded static SQL statements. During an implicit bind, if an object does not exist or a privilege has been revoked but not regranted, the execution of the package will fail. SYSCAT.PACKAGES.VALID is marked **N** when an index is dropped. The package will be implicitly rebound and VALID will equal **Y** even if the index has not been re-created. SYSCAT.PACKAGES.VALID is marked **N** when a table or view is dropped. The package will be implicitly rebound and VALID will equal **Y** only if the table or view has been re-created before the implicit rebind.

During BIND you can also specify the REOPT parameter. NONE is the default. You can select ALWAYS or ONCE, and each for the first time that a statement containing host variables, parameter markers, or special registers is executed, the values for these variables are used to optimize the access path for the statement.

SYSCAT.PACKAGE.VALID will be marked **X** if the package is inoperative because some function instance that it depends on has been dropped. This would occur if the package were dependent on a user-defined function and that user-defined function were dropped. These packages will **NOT** be implicitly rebound.

**Note**

User-defined functions are discussed in the class, *DB2 for Linux, UNIX, and Windows Database Advanced Programming*.

Dropping or creating a trigger on a column or table referenced by an application will cause the package to become invalid.

Explicit rebind



```
REBIND [PACKAGE] package-name
```

- New indexes available
- Statistics in catalog have changed
- Control when invalid packages are bound
- Must explicitly bind inoperative packages

WHY?

© Copyright IBM Corporation 2007

Figure 10-23. Explicit rebind

CF238.3

Notes:

Rebinding is the process of re-creating a package for an application program that was previously bound.

Invalid packages are automatically (or implicitly) rebound by the database manager when they are executed.

Inoperative packages must be explicitly bound again.

You may choose to rebind a package to take advantage of a newly created index or make use of updated statistics after executing the RUNSTATS command.

Another reason to explicitly rebind is to choose the time when the binding operation occurs, rather than allowing it to occur when a user requests the application that has an invalid package.

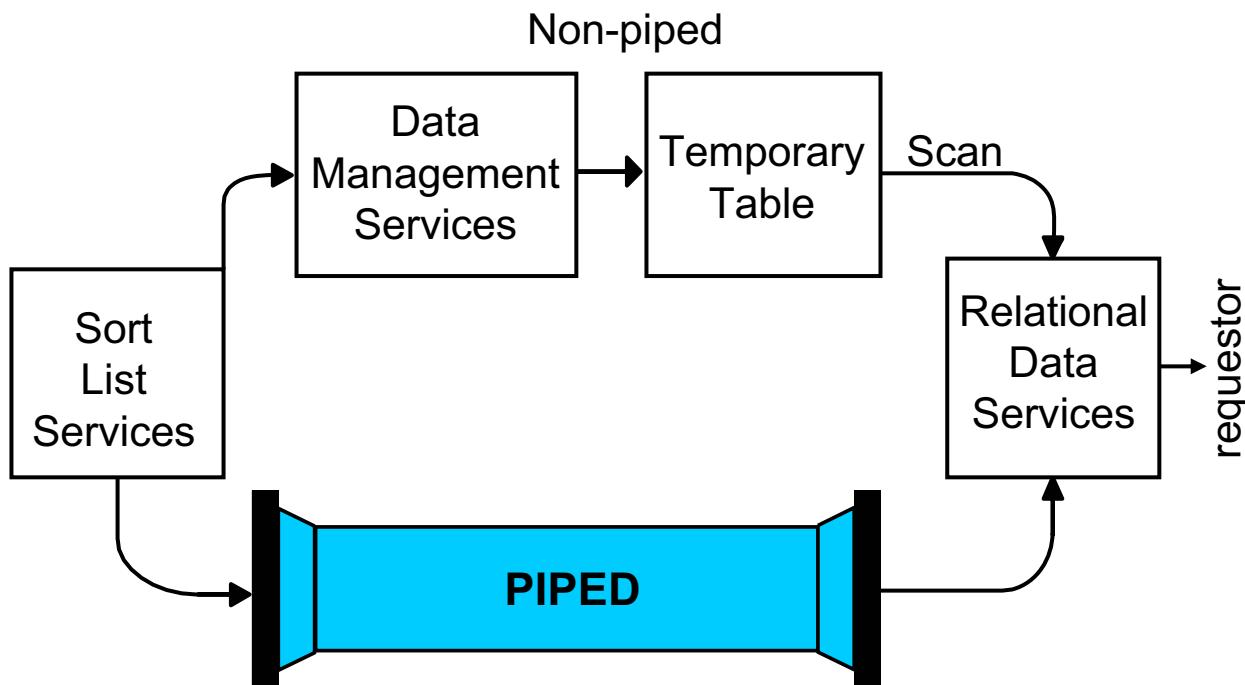
The authorization checking for a REBIND is based on the ID contained in the BOUND BY column of SYSCAT.PACKAGES.

Parameters cannot be changed through a REBIND operation.

With the db2rbind command you are able to rebind all packages of a given database.

References — Command Reference

Sorts — piped versus non-piped

© Copyright IBM Corporation 2007

Figure 10-24. Sorts — piped versus non-piped

CF238.3

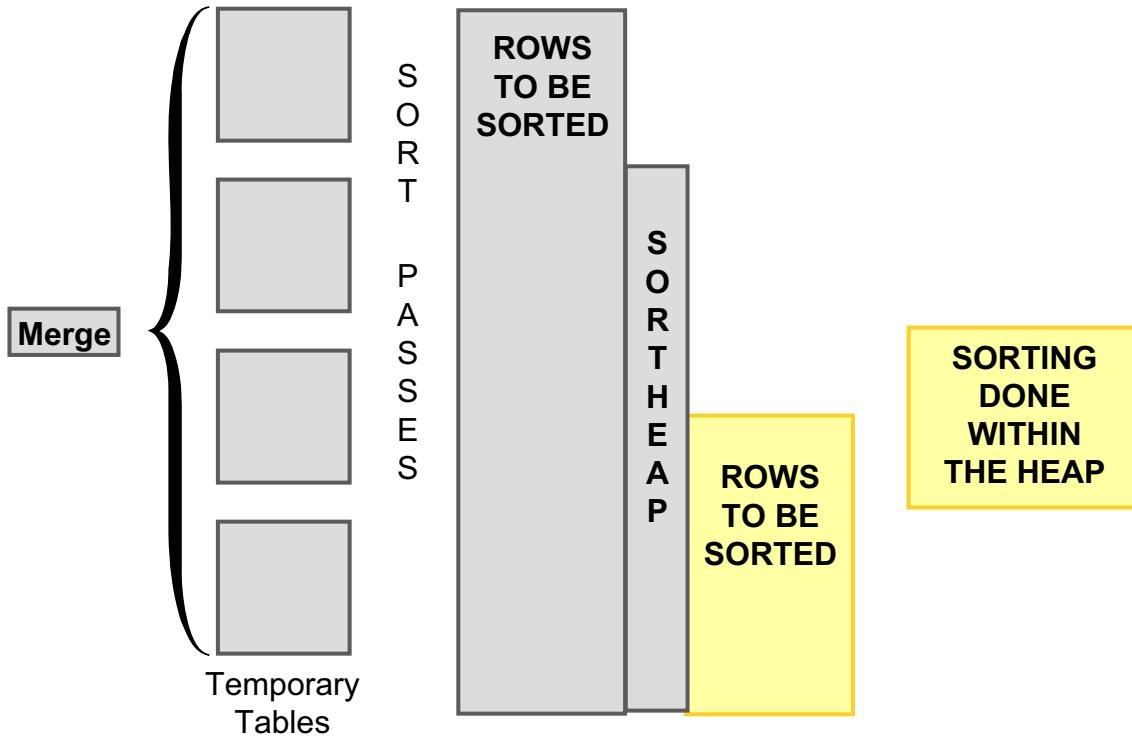
Notes:

A sort is piped if the rows can be returned directly from the sort buffers to Relational Data Services. Non-piped sorts require calls to Data Management Services in order to build a temporary table. This temporary table will then be scanned by Relational Data Services. Relational Data Services provides the results to the requesting application.

Piped sorts can provide performance benefits by reducing I/O operations. The number of piped sorts requested and the number accepted can be monitored. If many requested piped sorts are being rejected, tuning the database configuration parameter **sortheap**, or the database manager configuration parameter **sheapthres**, or both, may be necessary. **sortheap** determines the amount of memory allocated for an agent when a sort is requested. In-memory sorts are better performers than those that require sort passes. (See the next visual.)

sheapthres can be used to control the memory resources used to support sorting across all databases. When the value specified is reached, subsequent sort heap allocation against any database will be attempted, but will be constrained. Such *post threshold sorts* will not be piped. This parameter should be modified if benchmarking indicates poor balancing between sort performance and memory usage.

Sorts — Sort passes



© Copyright IBM Corporation 2007

Figure 10-25. Sorts — Sort passes

CF238.3

Notes:

Regardless of whether a sort is piped, a key consideration concerning the performance of sorts is the number of rows sorted.

If the number of rows sorted exceeds the size of the sort heap, Sort List Services will use temporary tables to hold subsets of the data to be sorted. If these tables cannot fit in memory, they will be written to disk. This is the *worst case* scenario illustrated in the visual.

The temporary files will contain the subset of rows sorted in a sort pass. Therefore, after all sort passes are complete, the temporary disk files will need to be read and merged. This technique increases the I/O and CPU cost of the sort.

If the space available in the sort heap can accommodate the entire set of rows to be sorted, a single pass sort can be completed and the use of temporary disk files is avoided.

Concurrent application tuning introduction

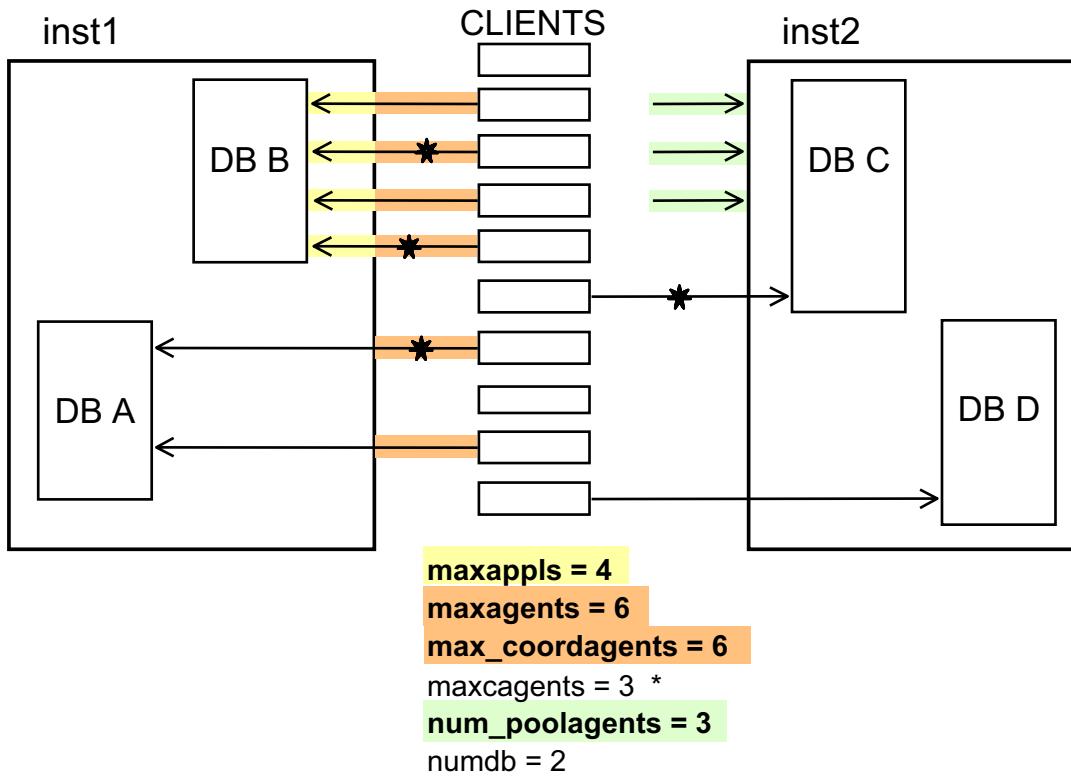


Figure 10-26. Concurrent application tuning introduction

CF238.3

Notes:

The values used in the above example are in no way intended to mean that such values are good starting points. Most likely, higher values will be used at your installation. The values used were chosen to illustrate the purpose of these parameters.

max_coordagents determines the maximum number of coordinating agents that can exist at one time on a server. **maxagents** controls the number of agents that can access any database within an instance. In a non-partitioned and non-SMP environment, **max_coordagents** and **maxagents** should be set to the same value. In the above example, **inst1** could not support another connection to the database **DBA** because **maxagents** has been reached. However, **inst2** could support additional agents.

maxappls serves a similar purpose as **maxagents**, but at the database level. It determines the maximum number of connections that can currently exist for the database. In the example, **DB B** could not support another connection because **maxappls** has been reached. However, **DB C** and **DB D** could support additional connections because **maxagents** and **maxappls** has not been reached.

maxcagents determines the number of agents that can simultaneously execute a transaction. Therefore, the value of this parameter is less than or equal to the value of maxagents. This parameter can be used to limit the amount of peak system resources necessary to support connected agents. While it does not prevent connections across the instance, (that is the function of maxagents), it does control simultaneous resource demands of the database. In the example, only three agents can be simultaneously executing a transaction. Therefore, inst1 would not allow any other currently connected application to execute an SQL statement. However, inst2 has only one agent currently executing SQL, so two additional agents could run transactions.

num_poolagents determines the number of agents that should be reserved in *standby* after they have been released by an application. In the example, inst2 has three idle agents waiting to become attached to a client when additional requests occur. Note that the idle agents, up to the number specified for the parameter, will exist either because they have previously served some application, or because they were created in the agent pool due to the setting of **num_initagents**.

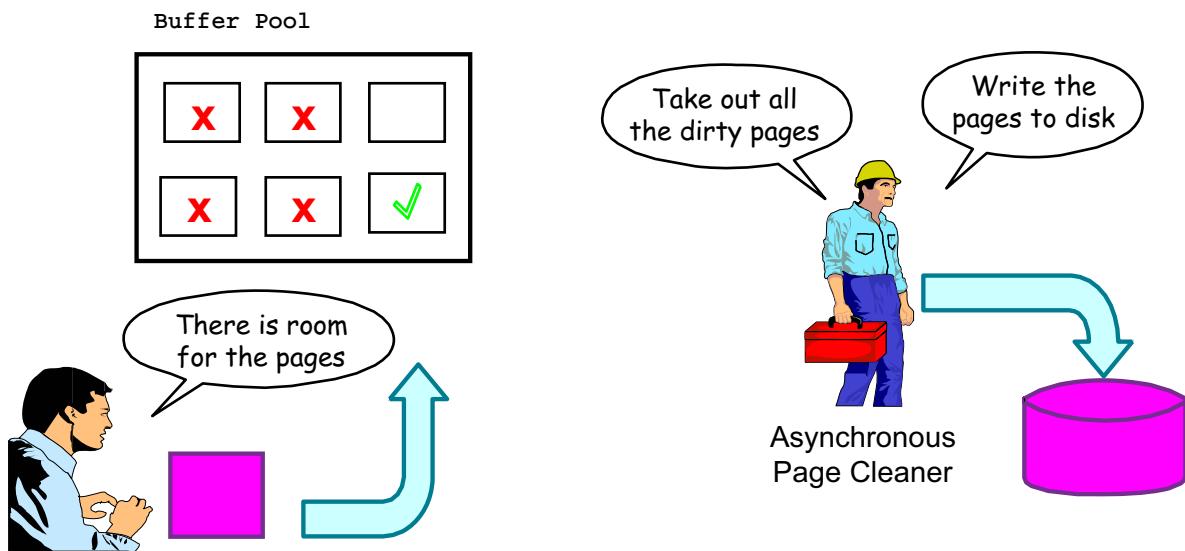
num_initagents determines the initial number of idle agents that are created in the agent pool at DB2START time.

numdb determines the number of active databases (those that can have one or more connections) within the instance. In the example, inst2 could not support a connection to a database other than DB C or DB D, even though the agent-oriented thresholds have not been reached.

Asynchronous Page Cleaner



CHNGPGS_THRESH=
NUM_IOCLEANERS=
BUFFPAGE=



© Copyright IBM Corporation 2007

Figure 10-27. Asynchronous Page Cleaner

CF238.3

Notes:

Asynchronous page cleaners will write changed pages from the buffer pool to disk before the space in the buffer pool is required by a database agent. This means that the agents will not have to wait for a changed page to be written out before being able to read a page, and your application's transactions should run faster.

Page cleaning is affected by the size of the buffer pool. The default size for buffer pools is set by the **BUFFPAGE** parameter, but the sizes of the independent buffer pools that are defined for the database is the primary setting that needs to be considered.

You may use the **CHNGPGS_THRESH** database configuration parameter to specify the level (percentage) of changed pages at which the asynchronous page cleaners will be started, if they are not currently active. When the page cleaners are started, they will build a list of the pages to write to disk. Once they have completed writing those pages to disk, they will become inactive again and wait for the next trigger to start.

In a read-only environment, these page cleaners are not used.

For databases with a heavy update transaction workload, ensure that there are enough clean pages in the buffer pool by setting the CHNGPGS_THRESH to be equal to or less than the default value (60). A percentage larger than the default can help performance if your database has a small number of very large tables.

The **NUM_IOCLEANERS** database configuration parameter allows you to specify the number of asynchronous page cleaners for a database. These page cleaners write changed pages from the buffer pool to disk before the space in the buffer pool is required by a database agent. This means that the agents will not wait for changed pages to be written out before being able to read a page. As a result, your application's transactions should run faster.

If you set the parameter to zero, no page cleaners are started and as a result, the database agents will perform all of the page writes from the buffer pool to disk. This parameter can have a significant performance impact on a database stored across many physical storage devices, since in this case there is a greater chance that one of the devices will be idle. If no page cleaners are configured, your applications may encounter periodic log full conditions.

If the applications for a database primarily consist of transactions that update data, an increase in the number of cleaners will speed up performance. Increasing the page cleaners will also decrease recovery time from soft failures, such as power outages, because the contents of the database on disk will be more up-to-date at any given time.

When setting the value for NUM_IOCLEANERS, consider the following recommendations:

- Application type
 - If it is a query-only database that will not have updates, set this parameter to zero.
 - If transactions are run against the database, set this parameter to be between one and the number of physical storage devices used for the database.
- Workload
 - Environments with high update transaction rates may require more page cleaners to be configured.
- Buffer pool sizes
 - Environments with large buffer pools may also require more page cleaners to be configured.

The database system monitor may be used to tune NUM_IOCLEANERS using information from the event monitor about write activity from a buffer pool.

10.5 Minimizing communication overhead

Minimizing communication overhead



"TALK ISN'T CHEAP"



- Techniques to minimize overhead:
 - Blocking
 - Stored procedures

© Copyright IBM Corporation 2007

Figure 10-28. Minimizing communication overhead

CF238.3

Notes:

In a client/server environment, communications can be a considerable cost element of applications.

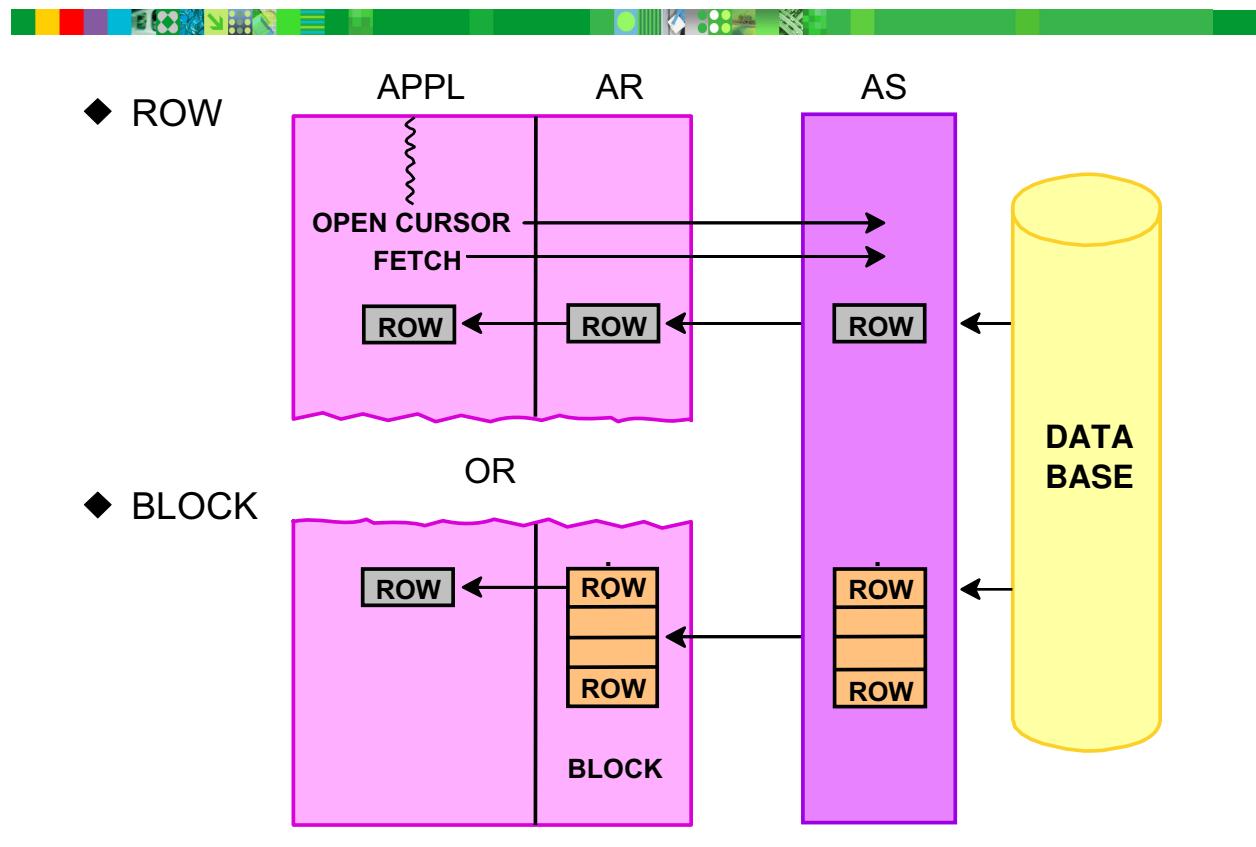
By reducing the amount of communication between the client and server, performance gains can be realized.

Techniques that may possibly be exploited are:

- Blocking
- Stored Procedures

These techniques are the subject of several subsequent pages.

Levels of data transmission



© Copyright IBM Corporation 2007

Figure 10-29. Levels of data transmission

CF238.3

Notes:

A non-blocking operation entails a high communication expense. Each **FETCH** request is sent from the client to the server, and one row is returned for each request. This technique is high in CPU and data transmission costs.

A blocking operation reduces the communication and CPU expense. When record blocking is utilized, a single **FETCH** request from the client will result in a multiple row transfer between the server and the client. The results are fetched one at a time from a buffer allocated at the client.

After the client buffer is emptied, the next **FETCH** will cause an additional block of rows to be returned from the server.

The application program logic is not changed to enable or utilize blocking, with the possible exception of cursor declarations. However, other factors will determine when blocking is utilized.

Database Manager configuration parameters that determine the size of the block are **rqrioblk** and **aslheapsz**. These parameters also determine the size of a communication

buffer between an agent and an application. For local applications, the parameter **aslheapsz** is used to allocate the cache for row blocking. For remote applications, the parameter **rqioblk** on the client workstation is used to allocate the cache for row blocking. The cache is allocated on the client. Generally, for adequate performance, these parameters must be greater than or equal to the following:

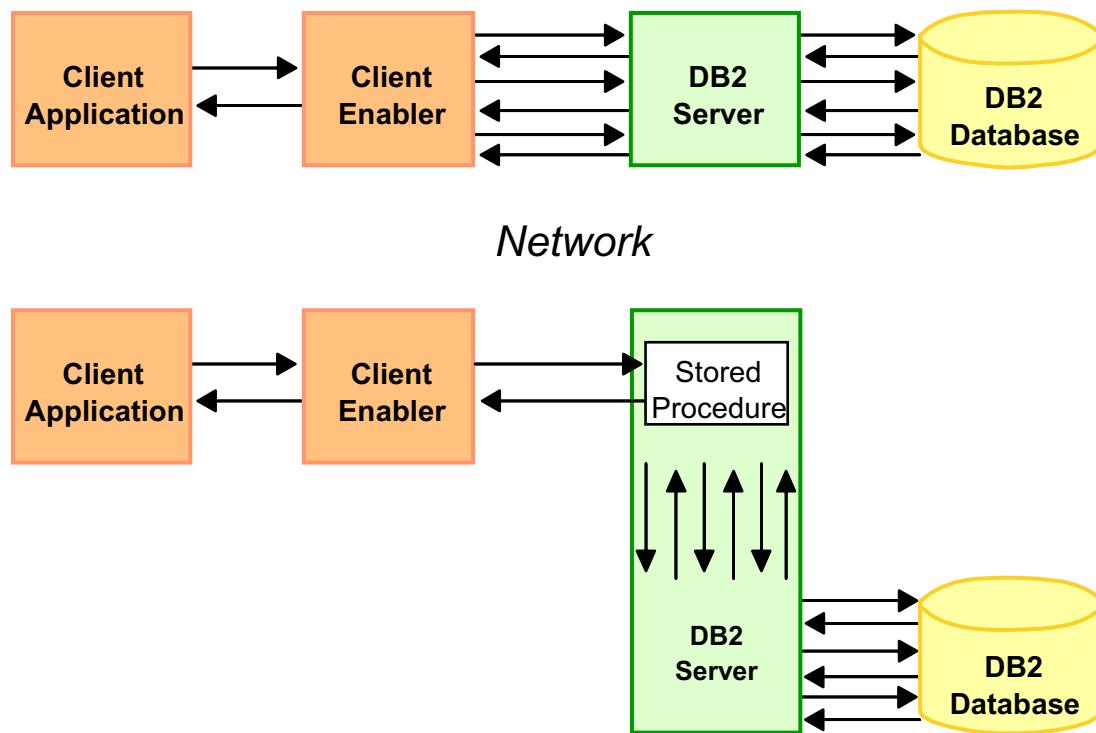
(sizeof (input SQLDA)+sizeof (output SQLDA)+sizeof (input SQLVARs)+250)

4096 (bytes/page)

Otherwise, the request to the database manager or the reply will not fit in a single send and receive pair. In simpler terms, the size of this buffer needs to accommodate the largest amount of data for a single request, or the request will be handled with more communication send/receive pairs.

If the system is not constrained on memory, setting these parameters to even larger values will cause larger blocks of memory to be allocated to support blocking cursors.

Stored procedures



© Copyright IBM Corporation 2007

Figure 10-30. Stored procedures

CF238.3

Notes:

Stored procedures allow an application running on a client to call a procedure stored on a database server. This *stored procedure* executes and accesses the database locally and returns information to the client application.

To use this technique, an application must be written in two separate procedures. The calling procedure is contained in a client application and executes on the client. The *stored procedure* executes at the location of the database on the database server.

Applications that use stored procedures have the following advantages:

- Reduced network traffic
- Improved performance of server-intensive work
- Access to features that exist only on the database server.
- Minimized maintenance if logic in the server code requires modification.

Stored procedures can be *fenced* or *not fenced*. Fenced procedures run in processes separate from the database agent process or thread. Not fenced procedures run as part of the agent process or thread. The advantage of not fenced procedures is performance,

since the stored procedure and database agent do not need to communicate through a router. However, the integrity of the database agent is exposed. Not fenced stored procedures must be carefully tested.

Database manager configuration parameters that impact fenced procedures are listed below:

- **fenced_pool** - For threaded db2fmp processes (processes serving threadsafe stored procedures and UDFs), this parameter represents the number of threads cached in each db2fmp process. For nonthreaded db2fmp processes, this parameter represents the number of processes cached.
- **keepfenced** - This parameter indicates whether a fenced mode process is kept after a fenced mode routine call is complete. Fenced mode processes are created as separate system entities in order to isolate user-written fenced mode code from the database manager agent process. This parameter is only applicable on database servers.

If *keepfenced* is set to *no*, and the routine being executed is not threadsafe, a new fenced mode process is created and destroyed for each fenced mode invocation. If *keepfenced* is set to *no*, and the routine being executed is threadsafe, the fenced mode process persists, but the thread created for the call is terminated. If *keepfenced* is set to *yes*, a fenced mode process or thread is reused for subsequent fenced mode calls.

When the database manager is stopped, all outstanding fenced mode processes and threads will be terminated.

Setting this parameter to *yes* will result in additional system resources being consumed by the database manager for each fenced mode process that is activated, up to the value contained in the *fenced_pool* parameter. A new process is only created when no existing fenced mode process is available to process a subsequent fenced routine invocation. This parameter is ignored if *fenced_pool* is set to 0.

- **num_initfenced** - This parameter indicates the initial number of nonthreaded, idle db2fmp processes that are created in the db2fmp pool at DB2START time. Setting this parameter will reduce the initial startup time for running non-threadsafe C and Cobol routines. This parameter is ignored if *keepfenced* is not specified.

It is much more important to set *fenced_pool* to an appropriate size for your system than to start up a number of db2fmp processes at DB2START time.

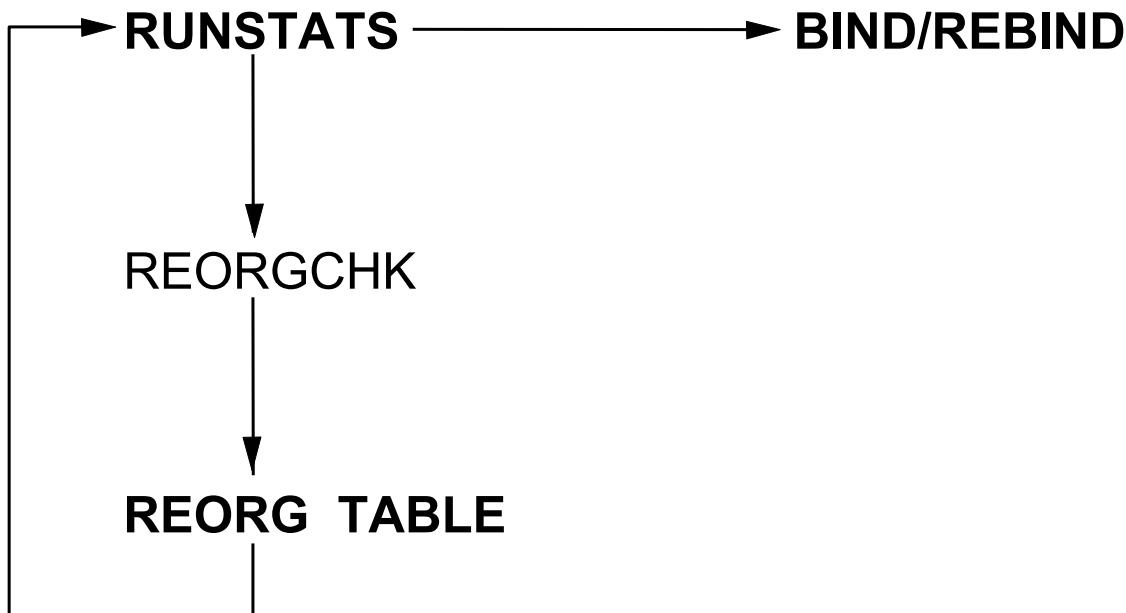
References

Application Programming Guide

Administration Guide

10.6 Utilities and tools

Operational utilities

© Copyright IBM Corporation 2007

Figure 10-31. Operational utilities

CF238.3

Notes:

RUNSTATS updates the statistics about tables and indexes. The statistics are stored in the system tables. RUNSTATS should be used after massive changes to the data and possibly after running REORG.

REORGCHK examines the data in the system tables and applies formulas to determine whether reorganization of a table, its indexes, or both is potentially needed. REORGCHK can also invoke RUNSTATS before examining the statistics. REORGCHK may be run periodically, or when users notice a degradation in performance.

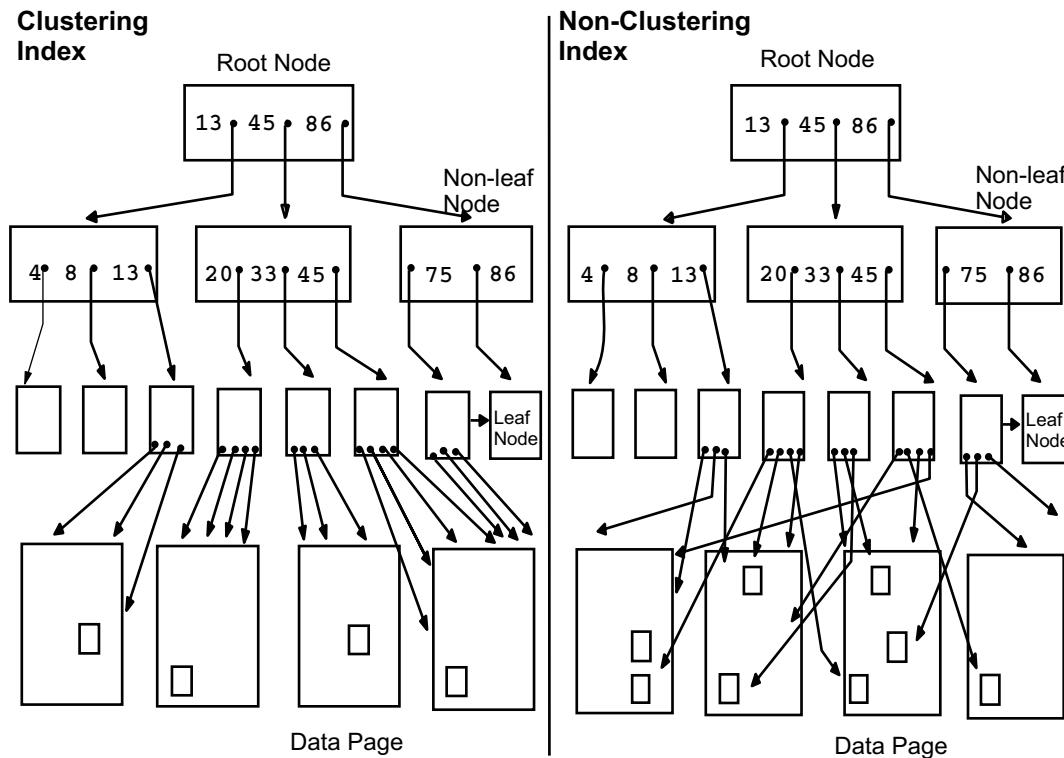
REORG eliminates fragmentation in tables and indexes and may optionally order the rows of a table according to the order of an index. REORG should be used when REORGCHK indicates that REORG is needed and when performance degrades over time: when data inserts, updates, and deletes cause the clustering or space utilization to degrade.

BIND of new applications should occur after statistics for the accessed tables and their supporting indexes have been gathered.

After using RUNSTATS, it MAY be advantageous to REBIND applications so that the DB2 optimizer can take advantage of the newly updated statistics when choosing the access paths for the programs. The degree to which you choose to REBIND depends on your knowledge of the changes that have occurred to the state of the tables accessed since the last bind activity. Some installations will REBIND critical applications after updating statistics without much analysis simply because the cost of doing so is considered minimal. Other installations will carefully monitor changes in statistics and only REBIND applications when some percentage of change to the statistics reaches a installation defined limit.

The `db2rbind` command can be used to rebind a number of packages in a database with one command. By default, it will only rebind invalid packages, but you can specify that rebinding of all valid and invalid packages is to be done. Documentation on `db2rbind` can be found in the *Command Reference*.

Clustered and non-clustered indexes



© Copyright IBM Corporation 2007

Figure 10-32. Clustered and non-clustered indexes

CF238.3

Notes:

An index is structured as a B+ tree. This structure enables the database manager to find values in the index rapidly.

The root node is at the top of this structure. In the illustration, the root node would contain an entry for each non-leaf, or intermediate node. The entry in the root node consists of the high value contained on the intermediate node and a pointer to this node.

The intermediate nodes are similar in structure to the root node, except that the range of values addressed is more specific. The intermediate node contains an entry for each of the leaf nodes addressed. The entry consists of the high value contained on the leaf node and a pointer to this leaf node.

The leaf nodes contain the value/rid pairs themselves. The leaf nodes collectively address the entire table.

The visual above is only one example of index structure. It is possible that an index contains more intermediate levels. Such an index would be required if the number of leaf nodes could not be addressed by one level of intermediate nodes.

Some insight regarding the structure of an actual index can be obtained by examining SYSCAT.INDEXES. The column NLEVELS indicates the number of root, intermediate, and leaf node levels. The column NLEAF indicates the number of leaf pages.

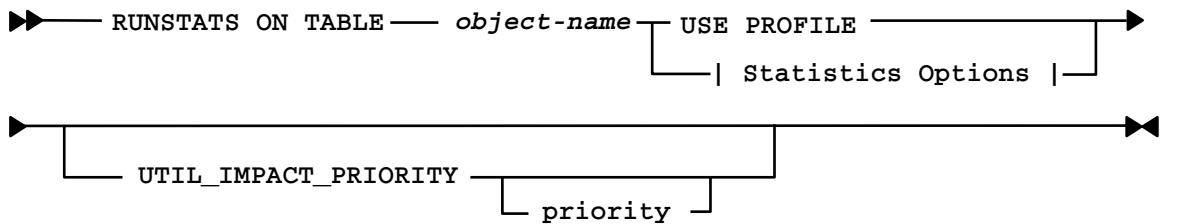
The degree to which the data rows correspond in order to the values on the leaf nodes is regarded as the clustering of the data. Therefore, the concept of clustering is only applicable when discussing a table and a given index.

The index on the left side of the visual illustrates a table clustered by an index, because the rows in the data pages are physically ordered according to the index sequence. The table on the right is not clustered by an index. The rows in the data pages do not seem to correspond to the order of the index entries.

During REORG, an index can be identified by which to cluster the data. Clustering benefits range type searches, such as BETWEEN or EQUAL for a non-unique key. This is due to a reduction in I/O costs. If the name of an index is not specified on the REORG command, and a clustering index exists, then the data will be ordered according to the clustering index.

INSERT, UPDATE, and DELETE activity can change the degree of clustering. For a clustering index, new rows are inserted physically close to rows where the key values of clustering index are in the same range. The CLUSTERRATIO or CLUSTERFACTOR columns of SYSCAT.INDEXES will indicate the degree of clustering for each index defined on the table.

RUNSTATS command syntax (basic)



Statistics Options:

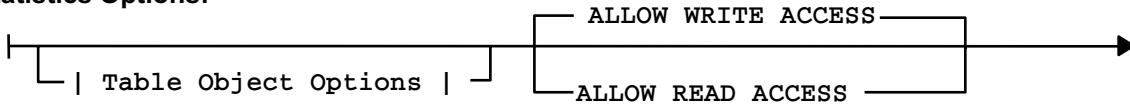
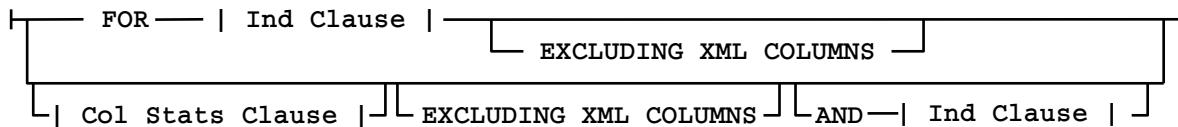


Table Object Options:



See [Command Reference](#) for more detailed RUNSTATS information.

© Copyright IBM Corporation 2007

Figure 10-33. RUNSTATS command syntax (basic)

CF238.3

Notes:

You must have SYSADM, SYSCTRL, SYSMAINT, DBADM, or CONTROL privilege on the table to invoke RUNSTATS.

If no options are specified, only basic table statistics will be updated, and other users will have access to the table while the statistics are being gathered.

The table name can be an alias or a fully qualified reference of the form **schema.tablename**.

FOR Index Clause indicates to update statistics for indexes only. If no table statistics have been previously collected on the table, basic table statistics are also collected. **AND Index Clause** indicates to update statistics for the indexes as well as the table.

Index Clause allows you to identify one, multiple, or ALL indexes on the table. The fully qualified name in the form of schema.index-name must be used.

DETAILED can be specified to calculate extended index statistics. **DETAILED** requests that **CLUSTERFACTOR** be determined as well as **PAGE_FETCH_PAIRS**. **CLUSTERRATIO** will not be determined. The **CLUSTERFACTOR** and **PAGE_FETCH_PAIRS** statistics in

combination reflect the clustering of the table to the index with a greater degree of accuracy than CLUSTERRATIO.

Using options with the Column Stats Clause, you can specify that statistics collection should be done on some columns but not on others. Columns can be specified either for basic statistics collection (on-cols-clause) or in conjunction with the WITH DISTRIBUTION clause (on-dist-cols-clause). The number of values collected is determined by the **numfreqval** database configuration parameter. The number of quantiles collected is determined by the **numquantiles** database configuration parameter. These statistics can be viewed through SYSCAT.COLDIST.

Distribution statistics are useful dynamic SQL or static SQL applications that do not use host variables, and where the SQL is accessing tables that lack relative uniformity in the data. Static SQL using host variables, or dynamic SQL using parameter markers, do not use distribution statistics.

ALLOW WRITE ACCESS specifies that other users can have read- and write-only access to the table while the statistics are being gathered. This is the default.

ALLOW READ ACCESS specifies that other users can have read-only access and write to the table while the statistics are being gathered.

If a table has been modified many times (lots of inserts, deletes, or updates) or has been reorganized with a REORG TABLE, it is a candidate for RUNSTATS.

If an application is performing poorly, it may be advantageous to REBIND the application **after** invoking RUNSTATS.

The syntax below gathers table statistics with distribution and detailed index statistics. (This is an example of gathering the greatest possible amount of statistics.)

```
RUNSTATS ON TABLE schema.tablename
WITH DISTRIBUTION
AND DETAILED INDEXES ALL
```

RUNSTATS can be invoked from the Control Center:

- Right-click and select table-name from the *Contents* pane.
- Select **Run Statistics...** from the pop-up menu.
- Choose the options for table, indexes and share level desired from the Run Statistics Window.
- Click **OK** to start the command.

If you want to see or save the command, or both:

- Select **Show Command** after choosing the options.

There is also another possibility to let DB2 do the runstats for you. In the Database Configuration file is a parameter called AUTO_RUNSTATS which is OFF by default. Setting this parameter to ON enables the automatic table runstats. A runstats policy (a defined set of rules or guidelines) can be used to specify the automatic behavior. To be able to use this option, you must turn the AUTO_MAINT parameter to ON.

All automatic maintenance features can also be configured from the Control Center. There you select (highlight) the database, right-click and select **Configure Automatic Maintenance**.

Determine if table reorganization is required

- Default is to call RUNSTATS first

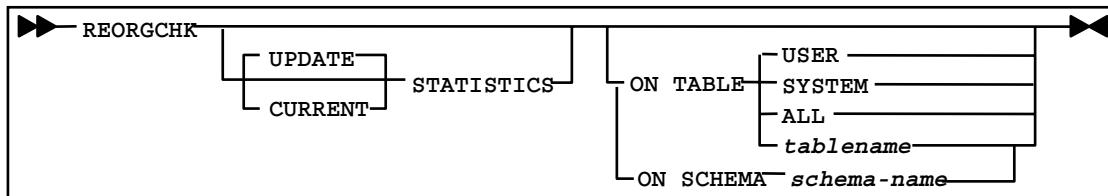


Table Statistics

SCHEMA.NAME	CARD	OV	NP	FP	ACTBLK	TSIZE	F1	F2	F3	REORG
<hr/>										
Table: INST01.ACT	18	0	1	1	-	648	0	- 100	---	

INDEX Statistics

Exceeded
Set bounds

SCHEMA.NAME	INDCARD	LEAF	ELEAF	LVLS	NDEL	KEYS...	F4	F5	F6	F7	F8	REORG
Table: INST01.ACT												
Index: INST01.PK_ACT	18	1	0	1	0	18...	100	-	-	0	0	
Index: INST01.XACT2	18	1	0	1	0	18...	100	-	-	0	0	

© Copyright IBM Corporation 2007

Figure 10-34. Determine if table reorganization is required

CF238.3

Notes:

To invoke REORGCHK, the user must have SYSADM, SYSCTRL, SYSMAINT, DBADM, or CONTROL privilege on the table.

UPDATE STATISTICS specifies to call the RUNSTATS routine to update table statistics and then use the updated statistics in the formula to determine if table reorganization is recommended. This is the default. **Running statistics on many tables can take time, especially if the tables are large.**

CURRENT STATISTICS specifies to use the current table statistics to determine if table reorganization is required.

USER specifies to check the tables that are under your current authorization ID. This is the default.

ALL specifies to check all user and system tables.

SYSTEM specifies to check the system tables only.

tablename can be explicitly provided in the form of an alias or a fully qualified reference of the form schema.tablename.

SCHEMA specifies a schema name to check all tables with that schema name.

If a table or its indexes exceeds the bounds of the formulas, then the table is a candidate for reorganization (REORG).

The Table Statistics Report headings mean:

- **CARD** - Number of rows in base table. In the example, the cardinality of the table is 1 million rows.
- **OV** - Number of OVERFLOW rows.
- **NP** - (NPAGES) number of pages that contain data.
- **FP** - (FPAGES) total number of pages.
- **TSIZE** - Table size in bytes. Calculated as the product of the number of rows in the table (CARD) and the average row length. The average row length is computed as the sum of the average column lengths (AVGCOLLEN in SYSCAT.COLUMNNS) plus 10 bytes of row overhead. For long fields and LOBs, only the approximate length of the descriptor is used. The actual long field or LOB data is not counted in TSIZE. For example, on the graphic, the table size is indicated as “1.10e+008” or 1.10×10^8 , or 110,000,000 bytes (about 110M).
- **F1, F2, F3** - Results of formulas 1, 2, and 3.
- **REORG** - Each hyphen (-) displayed in this column indicates that the calculated results were within the set bounds of the corresponding formula, and each asterisk (*) indicates that the calculated results exceeded the set bounds of its corresponding formula. “-” or “**” on the left side corresponds to F1 (Formula 1); “-” or “**” in the middle of the column corresponds to F2 (Formula 2); “-” or “**” on the right side of the column corresponds to F3 (Formula 3). Table reorganization is suggested when the results of the calculations exceed the bounds set by the formula.

The Index Statistics Report headings mean:

- **CARD** - Number of rows in base table.
- **LEAF** - Total number of index leaf pages.
- **ELEAF** - Number of pseudo empty index leaf pages. A pseudo empty index leaf page is a page on which all the RIDs are marked as deleted, but have not been physically removed.
- **LVLS** - (LEVELS) number of index levels.
- **ISIZE** - Index size, calculated from average column length for all columns participating in the index.
- **NDEL** - Number of pseudo deleted RIDs. A pseudo deleted RID is a RID that is marked deleted. This statistic reports pseudo deleted RIDs on leaf pages that are not pseudo empty. It does not include RIDs marked as deleted on leaf pages where all the RIDs are marked deleted.
- **KEYS** - (FULLKEYCARD) number of unique index entries. For example, on the graphic, the number of keys of ACCT_GRP is indicated as 1000; the number of keys of ACCTIDX is indicated as “1e+006”, or 1 million entries.
- **F4, F5, F6, F7, F8** - Results of formulas 4, 5, 6, 7, and 8.

- **REORG** - Each hyphen (-) displayed in this column indicates that the calculated results were within the set bounds of the corresponding formula, and each asterisk (*) indicates that the calculated results exceeded the set bounds of its corresponding formula. “-” or “**” on the left column corresponds to F4 (Formula 4); “-” or “**” in the second from left column corresponds to F5 (Formula 5); “-” or “**” in the middle column corresponds to F6 (Formula 6); “-” or “**” in the second column from the right corresponds to F7 (Formula 7); “-” or “**” on the right column corresponds to F8 (Formula 8). Index reorganization advice is as follows:
 - If the results of the calculations for Formula 1, 2, and 3 do not exceed the bounds set by the formula, and the results of calculations for Formula 4, 5, or 6 **do** exceed the bounds set, then index reorganization is recommended.
 - If only the results of the calculation Formula 7 exceed the bounds set, but the results of formulas 1, 2, 3, 4, 5, and 6 are within the set bounds, then cleanup of the indexes using the CLEANUP ONLY option of REORG INDEXES is recommended.
 - If the only calculation result to exceed the set bounds is that of Formula 8, then a cleanup of the pseudo empty pages of the indexes using the CLEANUP ONLY PAGES option of REORG INDEXES is recommended.

REORGCHK statistics

Table Statistics:

```
F1: 100 * Overflow / Card < 5%:  
F2: 100 * Table Size / ((FPages - 1) *  
    (TABLEPAGESIZE - 68)) > 70%  
F3: 100 * NPages / FPages > 80%
```

Index Statistics:

```
F4: Cluster Ratio or normalized Cluster Factor > 80%  
F5: 100 * ( KEYS*(LEAF_RECSEIZE+LEAF_RECSEIZE_OVERHEAD)+  
    (INDCARD-KEYS)*DUPKEYSIZE ) /  
    ( (NLEAF_NUM_EMPTY_LEAFS-1)*  
    (INDEXPAGESIZE-LEAF_PAGE_OVERHEAD) ) >  
    MIN(50,(100 - PCTFREE))  
F6: ( 100-PCTFREE ) * ( (FLOOR((100 - LEVEL2PCTFREE) / 100 *  
    (INDEXPAGESIZE - NLEAF_PAGE_OVERHEAD)/(NLEAF_RECSEIZE +  
    NLEAF_RECSEIZE_OVERHEAD))) * (FLOOR((100 - MIN(10,  
    LEVEL2PCTFREE))/100 * (INDEXPAGESIZE  
    NLEAF_PAGE_OVERHEAD)/(NLEAF_RECSEIZE +  
    NLEAF_RECSEIZE_OVERHEAD)) ** (NLEVELS - 3)) *  
    (INDEXPAGESIZE  
    LEAF_PAGE_OVERHEAD)) / (KEYS*(LEAF_RECSEIZE+LEAF_RECSEIZE_OVERHEAD)+  
    (INDCARD - KEYS) * DUPKEYSIZE ) ) < 100  
F7: 100 * (NUMRIDS_DELETED /  
    (NUMRIDS_DELETED +CARD)) < 20  
F8: 100 * (NUM_EMPTY_LEAFS/NLEAF) < 20
```

© Copyright IBM Corporation 2007

Figure 10-35. REORGCHK statistics

CF238.3

Notes:

REORGCHK performs calculations on the statistics in order to determine if the tables and their corresponding indexes can benefit from being reorganized.

The following formulas are used **to analyze the physical location of rows and the size of the table**:

- **F1** looks at the number of overflow rows in the table. The number of overflow rows in the table should be less than 5 percent of the total number of rows. Overflow rows can be created when rows are updated and the new rows contain more bytes than the old ones (for example, when updating VARCHAR fields and making them longer), or when columns are added to existing tables.
- **F2** looks at the table size in bytes. The table size should be more than 68 percent of the total space allocated for the table. There should be less than 32 percent free space. The total space allocated for the table depends upon the page size of the table space in which the table resides (minus an overhead of 76 bytes). Because the last page allocated is usually not filled, 1 is subtracted from FPAGES.

- For MDC tables:

```
100*TSIZE / ((ACTBLK-FULLKEYCARD) * EXTENTSIZE * (TABLEPAGESIZE-68)) >
70
```

FULLKEYCARD represents the cardinality of the composite dimension index for the MDC table. Extentsize is the number of pages per block. The formula checks if the table size in bytes is more than the 70 percent of the remaining blocks for a table after subtracting the minimum required number of blocks.

- **F3** looks at empty pages. The number of pages that contain no rows at all should be less than 20 percent of the total number of pages. Pages can become empty after rows are deleted.

- For MDC tables, the formula is:

```
100 * activeblocks / ( ( fpages / ExtentSize ) - 1 )
```

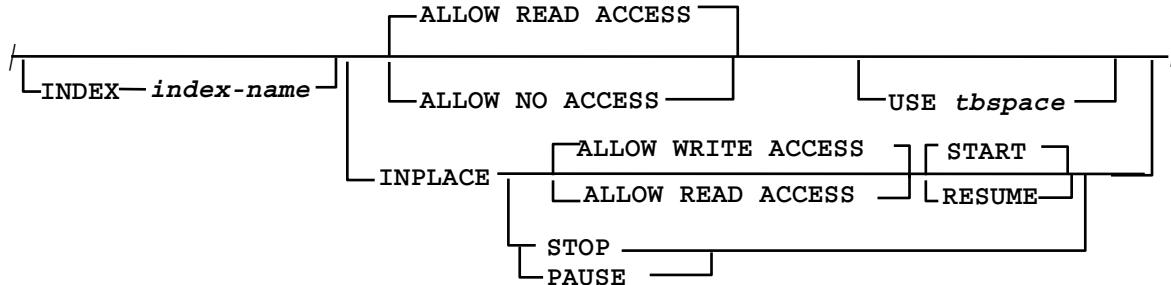
The following formulas are used to analyze the relationship of the indexes to the table data:

- **F4** looks at the cluster ratio of an index. The cluster ratio for the index by which you want the table data clustered should be greater than 80 percent. When multiple indexes are defined on a table, some of these indexes probably have a low cluster ratio. When looking at the output from REORGCHK, you want to check the value for F4 for the clustering index, or the index by which you want the data clustered. If the data is not well clustered according to the other indexes, this would not be a reason to perform a REORG.
- **F5** examines the empty space in an index. Less than 50 percent of the space reserved for index entries should be empty. This value is only checked when the number of leaf pages is greater than 1.
- **F6** looks at whether the index entries would fit into an index tree with one fewer level. This formula checks the ratio between the amount of space in an index tree that has one less level than the current tree, and the amount of space needed. If a tree with one less level could be created and still leave PCTFREE available, then a reorganization is recommended. The actual number of index entries should be more than 90% (or 100 - PCTFREE) of the number of entries an NLEVELS-1 index tree can handle (only checked if NLEVELS > 1).
- **F7** looks at how many pseudo deleted record identifiers there are on pages that also have valid record identifiers. The number of pseudo deleted RIDs on non-pseudo empty pages should be less than 20 percent. Entries are pseudo deleted in the index to minimize the requirement for next key locking to protect index entries.
- **F8** looks at the number of leaf pages that are empty due to pseudo deletes. The number of pseudo empty leaf pages should be less than 20 percent of the total number of leaf pages.

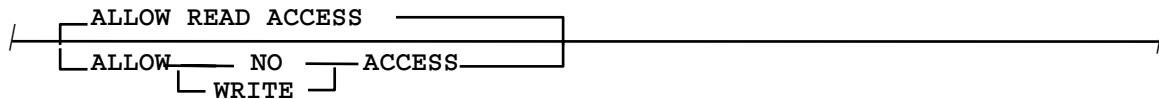
Reorganize table / index command syntax

```
► REORG [ TABLE —table-name— / Table Clause ] ... ►
  INDEXES ALL FOR TABLE —table-name— / Index Clause |
```

Table Clause:



Index Clause:



Consult the *Command Reference* for more detailed syntax for the REORG statement.

© Copyright IBM Corporation 2007

Figure 10-36. Reorganize table / index command syntax

CF238.3

Notes:

You must have SYSADM, SYSCTRL, SYSMAINT, DBADM, or CONTROL on table to REORG TABLE.

The REORG utility reorganizes a table or its indexes.

The table option reorganizes a table by reconstructing the rows to eliminate fragmented data, and by compacting information.

The index option reorganizes all indexes defined on a table by rebuilding the index data into unfragmented, physically contiguous pages.

The table to be reorganized can be specified through an alias or a fully qualified reference of the form schema.tablename. If you omit the schema name, the default schema is assumed.

Options on the command include the following:

- **INDEX index-name** specifies the index to use when reorganizing the table. If you do not specify the fully qualified name in the form: schema.index-name, the default schema is assumed. The schema is the user name under which the index was created. The

database manager uses the index to physically reorder the records in the table it is reorganizing.

For an inplace table reorg, if a clustering index is defined on the table and an index is specified, it must be a clustering index. If the inplace option is not specified, any index specified will be used. If you do not specify the name of an index, the records are reorganized without regard to order. If the table has a clustering index defined, however, and no index is specified, then the clustering index is used to cluster the table.

All of the indexes of the table are reorganized if the table is reorganized without the INPLACE option. None of the indexes of the table are reorganized if the INPLACE option is specified.

- **{ ALLOW READ ACCESS | ALLOW NO ACCESS}**

ALLOW READ ACCESS indicates that you want to allow only read access to the table during reorganization. This is the default behavior for reorganizations that do not specify INPLACE.

ALLOW NO ACCESS indicates that no one will be allowed access to the table during reorganization.

- **USE tbspace** specifies the name of a system temporary table space in which to store a temporary copy of the table being reorganized. If you do not provide a table space name, the database manager stores a working copy of the table in the table spaces that contain the table being reorganized.
- **INPLACE** reorganizes the table while permitting user access. The table is reorganized within its existing table space, using only a minimum amount of additional space to accomplish the reorganization. No index reorganization is accomplished during an INPLACE table reorg.
- **{ ALLOW WRITE ACCESS | ALLOW READ ACCESS }**

ALLOW WRITE ACCESS indicates that you want to allow write access to the table during reorganization. This is the default behavior for an INPLACE reorganization.

ALLOW READ ACCESS indicates that you want to allow only read access to the table during reorganization.

- **{ START | RESUME }**

START starts the inplace REORG processing. Because this is the default, this keyword is optional.

RESTART continues or resumes a previously paused inplace table reorganization.

- **{ STOP | PAUSE }**

STOP stops the inplace REORG processing at its current point.

PAUSE suspends or pauses the inplace REORG for the time being.

When the inplace reorganization is paused or stopped, the part of the table that has been processed to that point will have the benefits of a reorganized table (elimination of

fragmentation and compaction of data. A paused inplace reorganization can be resumed. A stopped inplace reorganization would need to be started again from the beginning to reorganize the entire table. You can pause an inplace reorganization for several reasons. For example, you may want to allow unimpeded use of the table by your online jobs (removing the overhead of the reorg from the system); you can then resume the utility when activity on the system is less intense. You may also pause an inplace reorganization as you prepare to stop the instance; the utility then can be resumed after the instance is restarted.

Use REORGCHK to determine if a table requires REORG.

After REORG, use RUNSTATS to update the table statistics.

If you have turned the Database Configuration Parameter AUTO_AMIN to ON you can also use the automatic REROG parameter AUTO_REORG and set it to on. This enables automatic Table and Index reorganization for the database. A reorganization policy (a defined set of rules or guidelines) can be used to specify the automatic behavior.

The automatic REORG function includes policies and includes things like:

- A state-based collection health indicator in the health monitor to surface any reorganization action that cannot be handled by the policy
- Notification of the identification of tables requiring manual reorganization
- Knowledge in DB2 to automatically identify the need for reorganization of a table
- Knowledge in DB2 to determine the type of reorganization required and when to run it
- Scheduling of reorganization action based on specifications

All this can also be configured using the Control Center GUI (Configure Automatic Maintenance) when selecting the database and use the right mouse button.

References

Command Reference

Statistical Catalog views



- **SYSSTAT.**
 - COLDIST
 - COLUMNS
 - COLGROUPS
 - COLGROUPTDIST
 - COLGROUPTDISTCOUNTS
 - INDEXES
 - ROUTINES (formerly FUNCTIONS)
 - TABLES

© Copyright IBM Corporation 2007

Figure 10-37. Statistical Catalog views

CF238.3

Notes:

The catalog views are defined so that statistics used by the optimizer can be updated through the view. An object will only meet the view definition for a given user if the user created the object, has control of the object, or holds the explicit DBADM authority.

With the exception of the ROUTINES view statistics, the normal technique to update statistics is to execute RUNSTATS. However, the updatable catalog views may be helpful in prototyping or problem determination.

COLDIST contains distribution statistics for the **n** most frequently occurring values within a column.

COLGROUPS contains information regarding the cardinality of groups of columns.

COLUMNS contains information concerning cardinality of the column as well as the range of values in the column.

INDEXES contains information regarding the number of levels and number of leaf pages in the index, the clustering of the index, distinct values within the index, and scan estimates for the index with varying size buffer pools.

ROUTINES contains information regarding cost estimates of User-Defined Functions, Stored Procedures, and Methods. I/O costs and CPU instructions are estimated in various classifications. These statistics are NOT updated by RUNSTATS and must be provided by the user if desired.

TABLES contains information regarding the number of rows in the table, the page utilization of the table, and the number of rows that have been relocated due to update activity.

You can run db2look or Generate DDL... from the Control Center to get a report on a table's statistics. From the Control Center:

- Right-click and select table-name from the *Contents* pane.
- Select **Generate DDL...** from the pop-up menu.
- Choose **Database statistics** and deselect other choices.
- Select **Generate to run the command**.
- Check the output in the Journal.

Consult the *Administration Guide* for details.

References

Administration Guide

The db2look mimic mode

- Using db2look to mimic the tables in a database
 - To extract the DDL for the tables in the database, use the **-e** option.
 - For example, create a copy of the SAMPLE database called SAMPLE2 such that all of the objects in the first database are created in the new database:

```
C:\>db2 create database sample2 DB20000I
The CREATE DATABASE command completed successfully.
C:\>db2look -d sample -e > sample.ddl
-- USER is:
-- Creating DDL for table(s)
-- Binding package automatically ...
-- Bind is successful
-- Binding package automatically ...
-- Bind is successful
```

- Edit the **sample.ddl** file
- Once you have changed the connect statement, execute the statements, as follows:

```
C:\>db2 -tvf sample.ddl > sample2.out
```

© Copyright IBM Corporation 2007

Figure 10-38. The db2look mimic mode

CF238.3

Notes:

There are many times when it is advantageous to be able to create a database that is similar in structure to another database. For example, rather than testing out new applications or recovery plans on a production system, it makes more sense to create a test system that is similar in structure and data, and to then do the tests against it instead. This way, the production system will not be affected by the adverse performance impact of the tests or by the accidental destruction of data by an errant application. Also, when you are investigating a problem (such as invalid results, performance issues, and so on), it may be easier to debug the problem on a test system that is identical to the production system.

You can use the db2look tool to extract the required DDL statements needed to reproduce the database objects of one database in another database. The tool can also generate the required SQL statements needed to replicate the statistics from the one database to the other, as well as the statements needed to replicate the database configuration, database manager configuration, and registry variables. This is important because the new database may not contain the exact same set of data as the original database but you may still want the same access plans chosen for the two systems.

The db2look tool is described in detail in the *DB2 Command Reference*, but you can view the list of options by executing the tool without any parameters. A more detailed usage can be displayed using the -h option.



Note

If you want the DDL for the user-defined spaces, database partition groups and buffer pools to be produced as well, add the "-l" flag after "-e" in the command above. The default database partition groups, buffer pools, and table spaces will not be extracted. This is because they already exist in every database by default. If you wish to mimic these, you must alter them yourself manually.

Mimicking statistics for tables

If the intent of the test database is to do performance testing or to debug a performance problem, it is essential that access plans generated for both databases are identical. The optimizer generates access plans based on statistics, configuration parameters, registry variables, and environment variables. If these things are identical between the two systems then it is very likely that the access plans will be the same.

Extracting configuration parameters and environment variables

The optimizer chooses plans based on statistics, configuration parameters, registry variables, and environment variables. As with the statistics, db2look can be used to generate the necessary configuration update and set statements. This is done using the -f option.



Note

Only those parameters and variables that affect DB2 compiler will be included. If a registry variable that affects the compiler is set to its default value, it will not show up under "Environment Variables settings".

10.7 Additional learning opportunities

Need more information? (1 of 3)



- CF46 — *DB2 for Linux, UNIX, and Windows Advanced Database Administration for Experts* (five days with labs)



- CF41 — *DB2 for Linux, UNIX, and Windows Performance Tuning and Monitoring Workshop* (four days with labs)

© Copyright IBM Corporation 2007

Figure 10-39. Need more information? (1 of 3)

CF238.3

Notes:

DB2 for Linux, UNIX, and Windows Advanced Database Administration for Experts - CF46

Perform advanced database administration tasks using DB2. These tasks include advanced load and utility issues, parallelism and Symmetric Multiprocessor (SMP) exploitation, DB2 Governor, job scheduling, job log, scripting, distributed data management, remote administration, advanced monitoring using the snapshot table functions, and federated database exploitation.

DB2 for Linux, UNIX, and Windows Performance Tuning and Monitoring Workshop - CF41

Learn how to tune for optimum performance the IBM DB2 for Linux, UNIX, and Windows relational database management system and associated applications written for this environment. Learn about DB2 for Linux, UNIX, and Windows on a serial processor, in a nonpartitioned database environment. Explore performance issues affecting the design of the database and applications using the database, the major database performance parameters, and the different tools that assist in performance monitoring and tuning.

More information (2 of 3)

- Programming education
 - CF10 — *DB2 for Linux, UNIX, and Windows Database Programming Workshop*
 - Two days with labs
 - Static SQL
 - CF11 — *DB2 for Linux, UNIX, and Windows Database Advanced Programming*
 - Three days with labs
 - Dynamic SQL, Stored Procedures, APIs, UDTs, UDFs
 - CG11 — *DB2 for Linux, UNIX and Windows Programming Using Java*
 - Four days with labs
 - SQLJ, Stored Procedures, JDBC, UDTs, UDFs
- Product documentation

© Copyright IBM Corporation 2007

Figure 10-40. More information (2 of 3)

CF238.3

Notes:

You, or other people in your organization, may desire programming details regarding the use of the application alternatives described in this unit.

DB2 for Linux, UNIX, and Windows Database Programming Workshop - CF10

Learn to produce application programs that manipulate DB2 databases in a Linux, UNIX, or Windows environment. Emphasis is on embedding SQL statements and preparing programs for execution.

Objectives:

- Connect to local and remote DB2 databases
- Incorporate static SQL statements in an application program
- Prepare the program for execution
- Discuss relational concepts
- Describe database locking from an application programming perspective
- Discuss program and DB2 options relative to performance of static SQL

DB2 for Linux, UNIX, and Windows Database Advanced Programming - CF11

Learn how to take advantage of advanced programming techniques to access DB2 databases in the workstation environment.

Objectives:

- Code programs to manipulate Large Objects (LOB)
- Code dynamic SQL applications
- Code stored procedures and call such procedures
- Provide programming support for user defined functions
- Use product defined Application Programming Interfaces (API)
- Code simple Call Level Interface (CLI) programs
- Identify factors that influence application performance of alternatives to static SQL

DB2 for Linux, UNIX and Windows Programming Using Java - CG11

Develop the skills you need to produce application programs in Java that manipulate DB2 databases. Learn how to use Java Database Connectivity (JDBC) to access data via SQL; perform updates, inserts, and deletes; and process data returned via SELECTs with result-set processing. Also, learn to use SQL for Java (SQLJ) to embed SQL statements into Java programs and prepare SQLJ programs for execution. Learn how to take advantage of advanced programming techniques to access DB2. Use DB2 for Windows in the labs, but the concepts presented apply to programming in Java for any DB2 database.

Objectives:

- Incorporate JDBC statements into an application program
- Connect to local and remote DB2 databases
- Discuss relational concepts
- Describe database locking from an application programming perspective
- Use object-relational capabilities of DB2, such as Large Object (LOB) manipulation, user-defined functions, and user-defined distinct types
- Code stored procedures and call such procedures
- Incorporate static SQL statements into an application program
- Prepare a program for execution
- Discuss program and DB2 options relative to performance of programs accessing DB2
- Identify tools that exist for monitoring the performance of the DB2 database and applications

More information (3 of 3)

- CF12 — *DB2 SQL Workshop*
 - Two days with labs
 - Basic SQL
- CF13 — *DB2 SQL for Experienced Users*
 - Two days with labs
 - DDL, complex selects
- CF71 — *DB2 Stored Procedures Programming Workshop*
 - Two days with labs
 - Stored Procedure Programming

© Copyright IBM Corporation 2007

Figure 10-41. More information (3 of 3)

CF238.3

Notes:

DB2 SQL Workshop - CF12

Learn about the Structured Query Language (SQL) and how it applies to the entire DB2 Family. This course is appropriate for individuals working in an OS/390, VM/VSE, AIX, Windows, or OS/2 environment. Complete extensive hands-on exercises to develop your skills to use the SQL you learn. Perform these exercises either by using the DB2 for UNIX, Windows, and OS/2 product in a Windows environment, or by using the DB2 for z/OS and OS/390, Version 7, product through a Time Sharing Option (TSO) interface to a z/OS system.

Objectives:

- Use (write) SELECT, INSERT, UPDATE, and DELETE statements
- Use simple CREATE TABLE and CREATE VIEW statements

DB2 SQL for Experienced Users - CF13

Learn how to code simple Data Definition Language (DDL) statements and complex SELECT statements. Learn to use outer joins, complex subqueries, recursive SQL, and

table expressions. Also, learn about features, such as triggers, user-defined types, and user-defined functions.

Objectives:

- Use advanced SQL constructs, such as recursive SQL, case expressions, check constraints, and triggers
- Discuss basic relational database concepts, such as referential integrity, tables, and indexes
- Create tables and indexes
- Use outer joins
- Use complex subqueries
- Use the major scalar functions
- Use views

DB2 Stored Procedures Programming Workshop - CF71

Learn how to recognize when an IBM DB2 stored procedure is the correct solution for an application, how to use the Stored Procedure Builder, how to CALL a stored procedure, how to create a stored procedure using Data Definition Language (DDL), and how to troubleshoot a stored procedure. This course includes lectures and machine lab exercises. The exercises run on a Windows NT workstation, but are applicable to all IBM DB2 platforms.

Objectives:

- Describe a stored procedure and justify its use in an application
- Understand the Stored Procedure Builder and its capabilities
- Describe the DB2 SQL Procedure Language (SQL PL) statements and how to use them in an application
- Describe the basic structure of a Java application using stored procedures
- Create a DB2 stored procedure using the SQL DDL statement CREATE PROCEDURE
- Describe troubleshooting approaches for stored procedures

Checkpoint

Exercise — Unit Checkpoint

- ___ 1. You are the only DBA permitted to create new packages on the production system. You are called at 2:00 a.m. by a member of the operations staff, who informs you that an application is ending with an SQLCODE of -805. What is your response?

- ___ 2. Assume that there is another DBA with the authority to BIND any existing application. Three nights after your previous call from the operations staff, you receive another call. This time, the SQLCODE is a -818. What would you advise the DBA on duty to do?

- ___ 3. Chris is a programmer who will automate the creation of backups for your DB2 system. In a meeting to determine the proper interface, she asks for your advice on how to proceed. What do you tell her?

- ___ 4. Mike is a programmer who will write an application to support part of a point-of-sales application. His program will need to check on the credit history of customers attempting to use a credit card. The input to the program is the credit card number. The credit history is in a 40-million row DB2 table, which has an index by credit card number. What interface would you suggest?

- ___ 5. Your consulting services are requested at a planning meeting of a major software vendor. The software product being discussed must be able to perform ad hoc queries against a number of different database products. In order to prevent having to provide specific code for each database product supported, you contribute a suggestion. What is it?

- ___ 6. The software application described in the prior question is going to be implemented using Microsoft Windows. What architecture might you recommend?

- ___ 7. As the database administrator, you have been constantly requested to supply samples of production data for a test machine. The test machine has enough security so that the content of the data can be a sampling of actual production data; however, only 5% of the data needs to be periodically sampled for the test environment. In the past, you have manually retrieved the sample and would like to eliminate this task from your busy schedule. You would like staff leaders from the programming group to be able to request a random 5% sampling of any table for

which they have SELECT authority. The application supporting this function would sample the data and optionally transmit the data to a target machine. How would you suggest implementing this application?

- ___ 8. A programmer is writing an application against a DB2 database. Some of the columns desired and the predicate is not totally known until execution time. What would you advise the programmer to use?

- ___ 9. You are attempting to determine the best cluster sequence for a large customer table. During your initial analysis, you have determined that online access to the table is generally via a unique CUSTNUMB. You also have many *batch* applications that require data to be sorted on LASTNAME. There are indexes on both of these columns. Given these brief requirements, what would you choose as the index to specify when performing a REORG, if either?

- ___ 10. The table analyzed for the prior question is examined more closely. Many online applications examine the data according to a range of ENTRY_DATE values. There is an index on ENTRY_DATE. Would you change your choice of cluster?

- ___ 11. An application programmer asks for your advice concerning a program that produces a simple business report from the table previously described. The program accesses the server from a client workstation and appears to run for longer than necessary. It simply FETCHES about 20% of the data and writes the rows to a report. EXPLAIN indicates that the index on ENTRY_DATE is being used to support a range predicate in the query. The programmer assures you that the same statement runs fine in CLP, but takes much longer inside the static program. What would be a PRECOMPILE / BIND parameter that you should examine as one, if not the first, possible cause of the problem?

- ___ 12. A programmer presents you with an application that has a very complex SQL statement that accesses many tables. Assume that the statement is logically correct and that your database design is proper. However, the programmer implies that the strategy externalized by EXPLAIN accesses the *wrong* table first. Assume that you have plenty of time and CPU for program preparation. What PRECOMPILE / BIND parameter might you experiment with to influence DB2?

- ___ 13. You have just finished analyzing the SQL in a package that has been identified as a performance problem. You determined that an index to support the application was required, so you created the index. Assume that the new index was also based on

the columns of the table by which the data should be clustered. What are your next three steps required for the package to take advantage of the new index?

Unit summary

Having completed this unit, you should be able to:

- Prepare applications that access DB2 data for execution
- Use PRECOMPILE and BIND options that are appropriate for specific application requirements
- Describe the application alternatives available to access DB2 data or request other DB2 functions
- List reasons for REBIND of application programs and execute this command
- Define the concept of single dimensional clustering, and determine the proper cluster sequence, if any, for DB2 tables
- Use RUNSTATS, REORGCHK, and REORG to enhance application performance

© Copyright IBM Corporation 2007

Figure 10-42. Unit summary

CF238.3

Notes:

Unit 11. Security

What this unit is about

This unit provides information about controlling access to DB2 objects through privileges using either explicit or implicit grants to group IDs and user IDs.

What you should be able to do

After completing this unit, you should be able to:

- Use DB2 access control mechanisms to implement security within the database
- Use group IDs to create a control hierarchy
- Describe Label Based Access Control (LBAC)
- Describe privileges within a database
- Describe privileges required for binding and executing a package
- Describe the difference between explicit privileges and implicit privileges
- Describe the different DB2 authorization levels

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

Command Reference

Administration Guide

Unit objectives



After completing this unit, you should be able to:

- Differentiate between authentication and authorization
- Use DB2 access control mechanisms to implement security within the database
- Use group IDs to create a control hierarchy
- Describe Label Based Access Control (LBAC)
- Explore privileges in the database
- Describe privileges required for binding and executing a package
- Describe the difference between explicit privileges and implicit privileges
- Describe the different DB2 authorization levels

© Copyright IBM Corporation 2007

Figure 11-1. Unit objectives

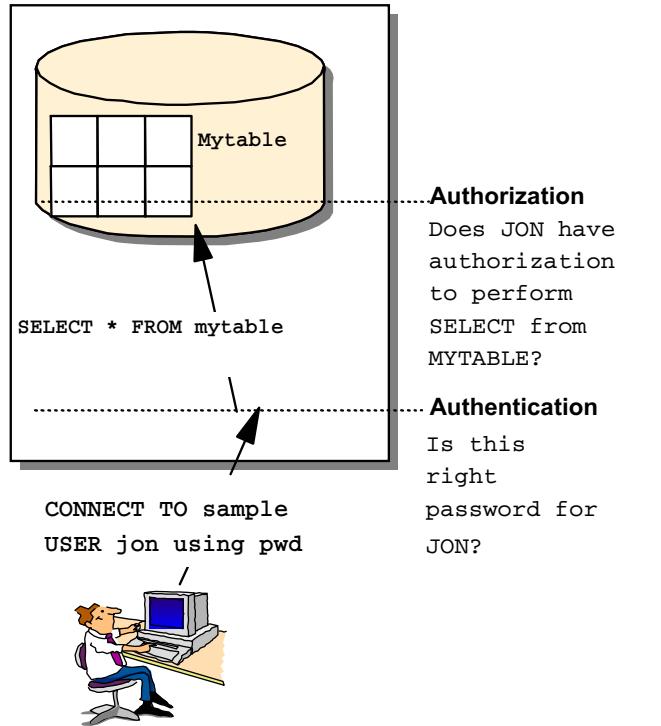
CF238.3

Notes:

11.1 Authentication versus authorization

Authentication versus authorization

- DB2 uses a combination of:
 - External security service
 - Internal access control information
- Authentication
 - Identify the user
 - Check entered username and password
- Done by security facility outside of DB2
- Authorization
 - Check if authenticated user may perform requested operation
 - Done by DB2 facilities
 - Information stored in DB2 catalog, DBM configuration file



© Copyright IBM Corporation 2007

Figure 11-2. Authentication versus authorization

CF238.3

Notes:

Access to an instance or a database first requires that the user be *authenticated*. The *authentication type* for each instance determines how and where a user will be verified. The authentication type is stored in the database manager configuration file at the server. It is initially set when the instance is created. There is one authentication type per instance, which covers access to that database server and all the databases under its control.

The following authentication types are provided:

SERVER — Specifies that authentication occurs on the server using local operating system security. If a user ID and password are specified during the connection or attachment attempt, they are compared to the valid user ID and password combinations at the server to determine if the user is permitted to access the instance. This is the default security mechanism.

**Note**

The server code detects whether a connection is local or remote. For local connections, when authentication is SERVER, a user ID and password are not required for authentication to be successful.

SERVER_ENCRYPT — Specifies that the server accepts encrypted SERVER authentication schemes. If the client authentication is not specified, the client is authenticated using the method selected at the server.

If the client authentication is SERVER, the client is authenticated by passing the user ID and password to the server. If the client authentication is SERVER_ENCRYPT, the client is authenticated by passing an encrypted user ID and encrypted password.

If SERVER_ENCRYPT is specified at the client and SERVER is specified at the server, an error is returned because of the mismatch in the authentication levels.

CLIENT — Specifies that authentication occurs on the database partition where the application is invoked using operating system security. The user ID and password specified during a connection or attachment attempt are compared with the valid user ID and password combinations on the client node to determine if the user ID is permitted access to the instance. No further authentication will take place on the database server. This is sometimes called single signon. If the user performs a local or client login, the user is known only to that local client workstation.

KERBEROS — Used when both the DB2 client and server are on operating systems that support the Kerberos security protocol. The Kerberos security protocol performs authentication as a third-party authentication service by using conventional cryptography to create a shared secret key. This key becomes a user's credential and is used to verify the identity of users during all occasions when local or network services are requested. The key eliminates the need to pass the user name and password across the network as clear text. Using the Kerberos security protocol enables the use of a single sign-on to a remote DB2 server.

Kerberos authentication works on Windows as follows:

1. A user logging on to the client machine using a domain account authenticates to the Kerberos key distribution center (KDC) at the domain controller. The key distribution center issues a ticket-granting ticket (TGT) to the client.
2. During the first phase of the connection, the server sends the target principal name, which is the service account name for the DB2 server service, to the client. Using the server's target principal name and the target-granting ticket, the client requests a service ticket from the ticket-granting service (TGS) which also resides at the domain controller. If both the client's ticket-granting ticket and the server's target principal name are valid, the TGS issues a service ticket to the client.

3. The client sends this service ticket to the server via the communication channel (which may be, as an example, TCP/IP).
4. The server validates the client's server ticket. If the client's service ticket is valid, then the authentication is completed.

It is possible to catalog the databases on the client machine and explicitly specify the Kerberos authentication type with the server's target principal name. In this way, the first phase of the connection can be bypassed.

If a user ID and a password are specified, the client will request the ticket-granting ticket for that user account and use it for authentication.

KRB_SERVER_ENCRYPT — Specifies that the server accepts KERBEROS authentication or encrypted SERVER authentication schemes. If the client authentication is KERBEROS, the client is authenticated using the Kerberos security system. If the client authentication is not KERBEROS, or the Kerberos authentication service is not available, then the system authentication type is equivalent to SERVER_ENCRYPT.



Note

The Kerberos authentication types are only supported on clients and servers running Windows 2000, Windows XP, and Windows.NET operating systems. Also, both client and server machines must either belong to the same Windows domain or belong to trusted domains. This authentication type should be used when the server supports Kerberos and some, but not all, of the client machines support Kerberos authentication.

Control login restrictions of connecting user on AIX server updated in V8.2

By default, when a user is authenticated on an AIX server, DB2 checks the connecting user's local login restrictions before allowing the connection to proceed. The DB2LOGINRESTRICTIONS registry variable permits DB2 to enforce alternative modes of login restrictions.

If DB2LOGINRESTRICTIONS is not set, the default value is LOCAL.

The variable may be set to the following values:

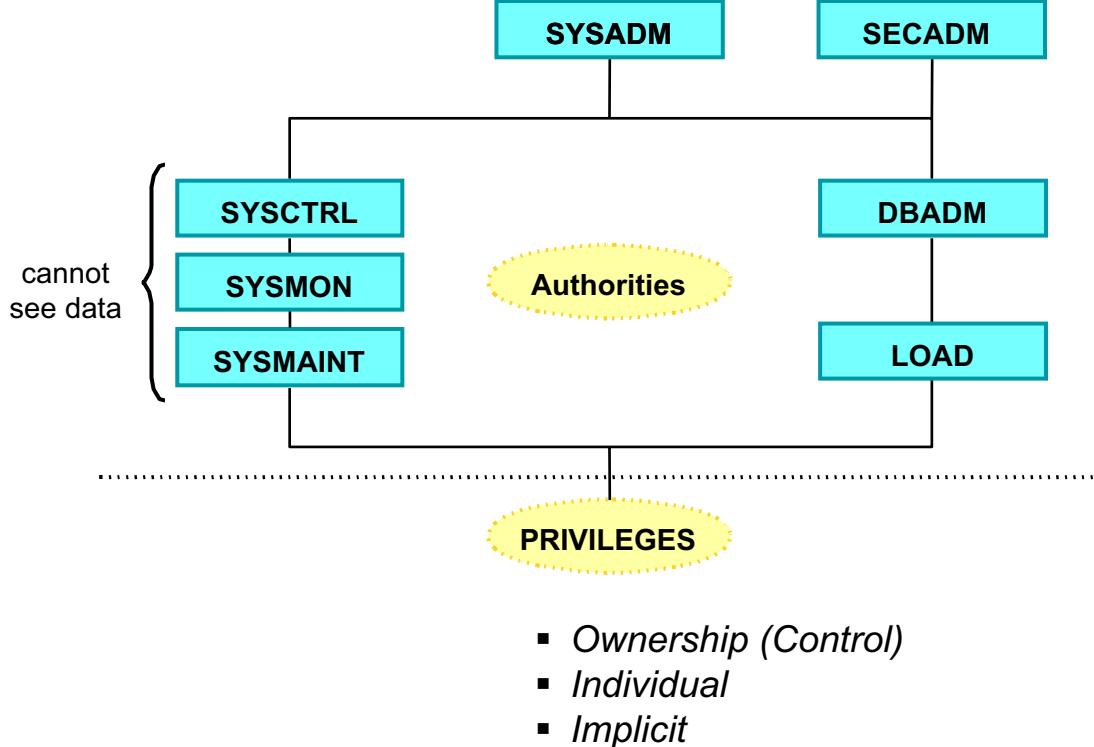
- REMOTE: DB2 will only enforce remote login restrictions
- SU: DB2 will only enforce su restrictions (su in AIX is changing the ID during a session)
- NONE: DB2 will not enforce any particular mode of login restrictions
- LOCAL: DB2 will only enforce local login restrictions

In all cases, DB2 still checks for the following error conditions:

- Expired account
- Locked account
- Invalid user

11.2 Access control — privileges

DB2 access control hierarchy



© Copyright IBM Corporation 2007

Figure 11-3. DB2 access control hierarchy

CF238.3

Notes:

This section describes the possible classes of users and their related authority levels in the DB2 environment.

When a privilege is associated with an object, that object must already exist.

Different levels of authority and several privileges can be assigned to users of DB2. Access to sysadm, sysctrl, sysmaint, and dbadm privileges should be restricted to avoid the risk of compromising system integrity. These authority levels are not required for general users.

Authority is a set of privileges covering a set of database objects. A user or group can have one or more of the following levels of authorization:

- **SYSADM** - Administrative authority - System Administrators are given full privileges over the entire DB2 instance. SYSADM can not be granted with a SQL statement.
- **SECADM** - The security administrator (SECADM) authority collects several security-related privileges under one authority. The abilities given by SECADM are not given by any other authority, not even SYSADM.

SECADM authority allows you to perform these actions:

- Create, drop, grant permission to access, or revoke the various objects that are part of label-based access control (LBAC)
- Use the TRANSFER OWNERSHIP statement on objects that you do not own
- Grant or revoke the SETSESSIONUSER privilege
- **SYSCTRL** - System Authority - System Controllers are given full privileges for managing the system, but are not allowed access to the data. SYSCTRL can not be granted with a SQL statement.
- **SYSMAINT** - System Authority - System Maintainers are given a subset of privileges for managing the system, but are not allowed access to the data. SYSMAINT cannot be granted with a SQL statement.
- **SYSMON** - Gives the ability to take database system monitor snapshots of a database manager instance or its databases. This includes the following commands: GET DBM MONITOR SWITCHES, GET MONITOR SWITCHES, GET SNAPSHOT, LIST ACTIVE DATABASES, LIST APPLICATIONS, LIST DCS APPLICATIONS, RESET MONITOR, UPDATE MONITOR SWITCHES.
- **DBADM** - Administrative authority - Database Administrators have control over an individual database. DBADM can be granted with a SQL statement. SYSCAT.DBAUTH has an entry for each database administrator.
- **LOAD** - The LOAD authority (LOAD) gives LOAD utility or AutoLoader utility privileges to load data into tables.

To create and set up an instance of the DB2 product, you must have operating system administration authority.

Privilege is the right of a particular user or group to create or access a resource.

1. **Ownership (CONTROL) privileges** - For most objects, the creator of the object has full access to that object. CONTROL privilege is automatically granted to the creator of an object (although some views are exceptions to this rule). CONTROL privileges are equivalent in all respects to ownership of the object, including the right to access an object in any way and to grant privileges on the object. Privileges are controlled by users with ownership or administrative authority. These users can grant or revoke privileges to or from other users.
2. **Individual privileges** - Individual privileges allow a specific function, sometimes on a specific object.
3. **Implicit privileges** - For example, users can be implicitly allowed to exercise privileges if they have the privilege to execute a package that requires those privileges.

On a UNIX platform, DB2 should be the only process running under the user ID of the instance owner. This safeguards the integrity of the operating system resources DB2 creates. If other processes were to run under this ID, they would have access to these resources, and might accidentally remove or corrupt them.

Database Authority summary



Function	SYSADM	SYSCTRL	SYSMAINT	DBADM	LOAD
MIGRATE DATABASE	YES				
UPDATE DBM CFG	YES				
GRANT/REVOKE DBADM	YES				
UPDATE db/node/dcs directories	YES	YES			
FORCE USERS OFF SYSTEM	YES	YES			
CREATE/DROP DATABASE	YES	YES			
CREATE/DROP/ALTER TABLE SPACE	YES	YES			
RESTORE TO NEW DATABASE	YES	YES			
UPDATE DB CFG	YES	YES	YES		
BACKUP DATABASE or TABLE SPACE	YES	YES	YES		
RESTORE TO EXISTING DATABASE	YES	YES	YES		
PERFORM ROLLFORWARD RECOVERY	YES	YES	YES		
START/STOP DATABASE INSTANCE	YES	YES	YES		
RESTORE TABLE SPACE	YES	YES	YES		
RUN TRACE	YES	YES	YES		
TAKE DBM or DB SNAPSHTOS	YES	YES	YES		
QUERY TABLE SPACE STATE	YES	YES	YES	YES	YES
UPDATE LOG HISTORY FILES	YES	YES	YES	YES	
QUIESCE TABLE SPACE	YES	YES	YES	YES	YES
REORG TABLE	YES	YES	YES	YES	
RUN RUNSTATS UTILITY	YES	YES	YES	YES	YES
READ LOG FILES	YES			YES	
CREATE/ACTIVATE/DROP EVENT MONITORS	YES			YES	

© Copyright IBM Corporation 2007

Figure 11-4. Database Authority summary

CF238.3

Notes:

System Administrator Authority - SYSADM authority is the highest level of administrative authority. Users with SYSADM authority can run utilities, issue database and database manager commands, and access the data in any table in any database within the database manager instance. It provides the ability to control all database objects in the instance, including databases, tables, views, indexes, and packages.

System Control Authority - SYSCTRL authority is the highest level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System control authority is designed for users administering a database manager instance containing sensitive data. A user with SYSCTRL authority has the implicit privilege to connect to a database.

System Maintenance Authority - SYSMAINT authority is the second level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations

can affect system resources, but they do not allow direct access to data in the databases. System maintenance authority is designed for users maintaining databases within a database manager instance that contains sensitive data. A user with SYSMAINT authority has the implicit privilege to connect to a database.

System Monitor Authority - SYSMON gives the ability to take database system monitor snapshots of a database manager instance or its databases. This includes the following commands: GET DBM MONITOR SWITCHES, GET MONITOR SWITCHES, GET SNAPSHOT, LIST ACTIVE DATABASES, LIST APPLICATIONS, LIST DCS APPLICATIONS, RESET MONITOR, UPDATE MONITOR SWITCHES.

Security administration authority - SECADM authority can only be granted by the SYSADM and can be granted to a user but not to a group. It gives these and only these abilities: Create and drop security label components, Create and drop security policies, Create and drop security labels, Grant and revoke security labels, Grant and revoke LBAC rule exemptions, Grant and revoke setsessionuser privileges, Execute the SQL statement TRANSFER OWNERSHIP on objects that you do not own.

No other authority gives these abilities, not even SYSADM.

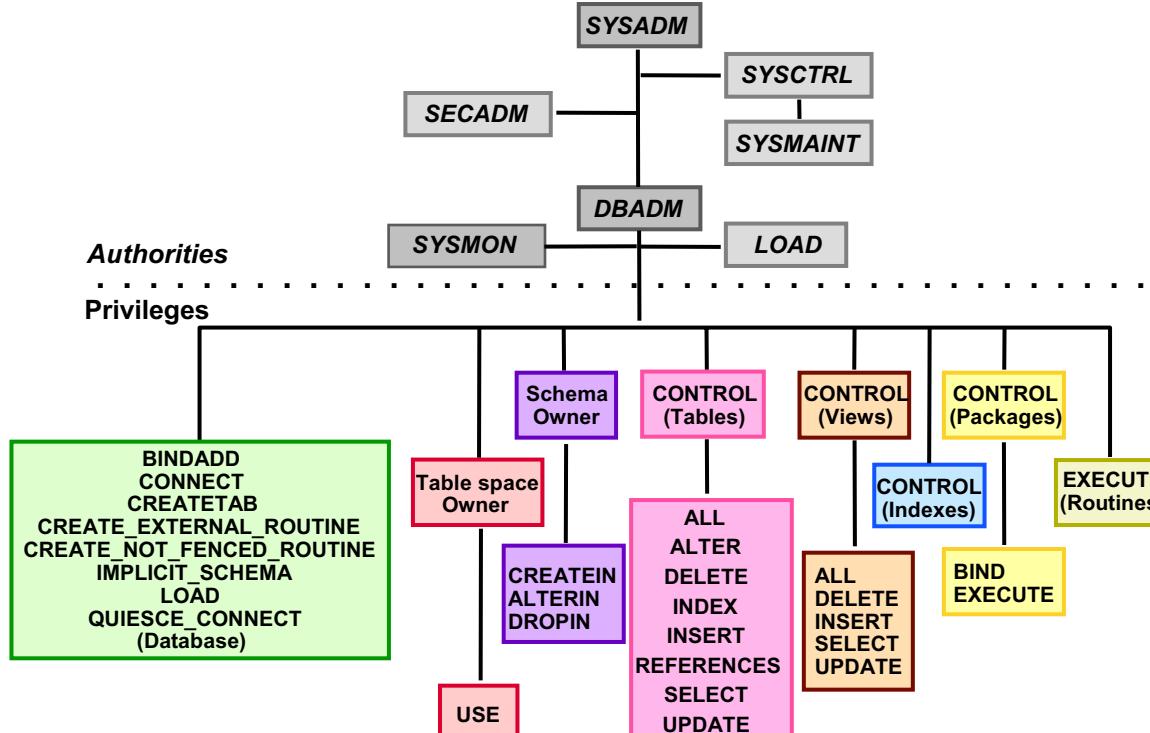
Database Administration Authority - DBADM authority is the second highest level of administrative authority. It applies only to a specific database, and allows the user to run utilities, issue database commands, and access the data in any table in the database. When DBADM authority is granted, BINDADD, CONNECT, CREATETAB, and CREATE NOT FENCED privileges are granted as well. Only a user with SYSADM can grant/revoke DBADM. Users with DBADM can grant privileges on the database to others and can revoke any privilege from any user regardless of who granted it.

A user may hold DBADM authority on more than one database, but a DBADM can only perform functions on the databases for which DBADM authority is held.

Load Authority - Users having LOAD authority at the database level, as well as INSERT privilege on a table, can use the LOAD command to load data into a table. Users having LOAD authority at the database level, as well as INSERT privilege on a table, can LOAD RESTART or LOAD TERMINATE if the previous operation was a load to insert data. If the previous load operation was a load replace, the DELETE privilege must also have been granted to that user before the user can LOAD RESTART or LOAD TERMINATE. If the exception tables are used as part of a load operation, the user must have INSERT privilege on the exception tables. The user with this authority can perform QUIESCE TABLESPACES FOR TABLE, RUNSTATS, and LIST TABLESPACES commands.

The database manager configuration parameter CATALOG_NOAUTH specifies whether users are able to catalog and uncatalog databases and nodes, or DCS and ODBC directories, without SYSADM authority. Default value is NO which means that SYSADM or SYSCTL authority is needed. If set to YES, the DB2 system administrator can allow users to maintain their own cataloging.

Authorities and privileges



© Copyright IBM Corporation 2007

Figure 11-5. Authorities and privileges

CF238.3

Notes:

Both DB2 authorities and privileges on objects are hierarchical in nature.

A **privilege** is the right of a particular user or group to create or access a resource.

Resources can be protected from unauthorized access by:

- Database privileges controlling both access and creation of databases.
- Table privileges controlling both access and creation of tables.
- View privileges controlling access and creation of views. Views can be used to control access to a table. By granting access to a view instead of the underlying tables, access is restricted to the rows and columns selected by the view definition.
- Index privileges controlling the creation of indexes.
- Package privileges controlling the creation, modification, and execution of packages.

Users with SYSAADM authority automatically possess the privileges associated with DBADM authority for each database in the instance.

DATABASE PRIVILEGES

- The creator of a database automatically receives DBADM authority.
- DBADM authority allows any SQL access to the database.
- DBADM authority is required to GRANT the following privileges:
 - BINDADD - Create new packages in the database.
 - CONNECT - Connect to the database.
 - CREATETAB - Create new tables in the database.
 - CREATE_EXTERNAL_ROUTINE - Allows a user to create a procedure for use by applications and other users of the database.
 - CREATE_NOT_FENCED_ROUTINE - Create a user-defined function or procedure that is *not fenced*. UDFs or procedures that are *not fenced* must be extremely well tested because the database manager does not protect its storage or control blocks from these UDFs. An *unfenced* UDF runs on the same side of the *firewall* as the database engine code.
 - IMPLICIT_SCHEMA - Create a schema implicitly by creating an object using a CREATE statement with a schema name that does not already exist. SYSIBM becomes the owner of the implicitly created schema, and PUBLIC is given the privilege to create objects in this schema.
 - LOAD - Load data into a table. LOAD is considered an authority since it implies other functions (quiesce table spaces, query table spaces, runstats), but it also gives the LOAD privilege at the database level.
 - QUIESCE_CONNECT - Allows a user to access the database while it is quiesced.
- Only users with SYSADM or DBADM authority can grant and revoke these privileges to and from other users.

TABLE SPACE PRIVILEGES

- The owner of the table space, typically the creator who has SYSADM or SYSCTRL authority, has USE privilege and the ability to grant this privilege to others.
- USE privilege allows the user to create tables within the table space.
- By default, at database creation time the USE privilege for table space USERSPACE1 is granted to PUBLIC, though this privilege can be revoked.
- The USE privilege cannot be used with SYSCATSPACE or any system temporary table spaces.

SCHEMA PRIVILEGES

- The owner of the schema has all of the schema privileges and the ability to grant them to others.
- Being owner of a schema allows GRANTing of:
 - ALTERIN - Alter objects in the schema.
 - CREATEIN - Create objects within the schema.
 - DROPIN - Drop objects within the schema.
- Objects that are manipulated within the schema object include: tables, views, indexes, packages, data types, functions, triggers, procedures, and aliases.

TABLE PRIVILEGES

- The creator of a table automatically receives **CONTROL** privilege. To grant CONTROL privilege on a table or view, the user must have SYSADM or DBADM authority.
- CONTROL privilege allows GRANTing of:
 - ALTER - Add columns to a table, add or change comments on a table and its columns, add a primary key or unique constraint, and create or drop a table check constraint. The user can also create triggers on the table, though additional authority on all objects referenced in the trigger, including SELECT on the table if the trigger references any columns of the table) is required. A user with ALTER on all the descendant table can drop a primary key; a user with ALTER privilege on the table and REFERENCES privilege on the parent table, or REFERENCES privilege on the appropriate columns, can create or drop a foreign key.
 - DELETE - Delete rows from the table/view.
 - INDEX - Create indexes on the table. The creator of the index has CONTROL privilege on index and can drop the index.
 - INSERT - Insert rows into the table. Run IMPORT utility against the table.
 - REFERENCES - Create referential constraints on a table, specifying the table as the parent in a relationship. The user might have this privilege only on specific columns.
 - SELECT - Retrieve rows from a table or view. Create a view on the table. Run the EXPORT utility against the table or view.
 - UPDATE - Change entries in the table or view or for one or more specific columns in a table or view. The user may have this privilege only on specific columns.
- CONTROL privilege is required to DROP the table, REORGANIZE the table, or refresh its statistics with RUNSTATS utility, and grant or revoke individual table privileges. When a user is granted CONTROL privilege on a table, all other privileges on that table are automatically granted WITH GRANT OPTION.
- The privilege to grant these privileges to others may also be granted using the WITH GRANT OPTION on the GRANT statement.

VIEW PRIVILEGES

- To create a view, SELECT or CONTROL privilege on every table or view referenced is required.
- CONTROL privilege is ONLY granted if the creator has CONTROL privilege on all tables and views referenced, or if they have SYSADM or DBADM authority.
- CONTROL privilege is required to grant DELETE, INSERT, SELECT, and UPDATE privileges; SELECT is the only option for **read-only** views.
- View privileges allow an authorized user to perform actions against base tables that they are not allowed to perform directly.

INDEX PRIVILEGES

- The creator of an index automatically receives CONTROL privileges.
- CONTROL privilege allows users to drop the index.
- No privilege is required to use an index.
- The table level INDEX privilege allows a user to create an index on that table.

PACKAGE PRIVILEGES

- The creator of a package automatically receives CONTROL privilege.
- CONTROL privilege allows granting of:
 - BIND - Bind or rebind the package and to add new package versions of the same package name and creator.
 - EXECUTE - Execute or run the package.
- CONTROL privilege implies that you can drop a package.
- The binder must have privileges or authority required to execute every embedded static SQL statement in the package granted to their userid or to PUBLIC.
- Any user with EXECUTE privilege may execute the statements within the scope of executing the package. They may not execute the statements in a dynamic environment (such as CLP) unless they have been granted authority to do so.
- Package privileges provide a powerful way to extend partial privileges to users through executing a package that contains embedded SQL.
- Database privilege BINDADD allows users to create new packages or rebind an existing package in the database.

ROUTINE PRIVILEGES

- Execute privileges involve actions on all types of routines such as functions, procedures, and methods within a database. Once having execute privilege, a user can then invoke that routine, create a function that is sourced from that routine (applies to functions only), and to reference the routine in any DDL statement such as CREATE VIEW, CREATE TRIGGER; or, when defining a constraint.
- The one who defines the externally stored procedure, function, or method receives EXECUTE WITH GRANT privilege.

SYSADM has implicit connect on all databases. DBADM has implicit connect on a specific database. When a database is created, the following privileges are automatically granted to PUBLIC. To remove these privileges, a DBADM or SYSADM must explicitly revoke the database privileges from PUBLIC.

- CREATETAB
- BINDADD
- CONNECT
- SELECT privilege on system catalog views.

Controlling use of schemas



- Named collection of objects
- Forms high-order part of objects with a two part name, for example, MEL.T1
- User with DBADM authority creates schema PAY for user MEL
`CREATE SCHEMA PAY AUTHORIZATION MEL`
- Mel can create objects in schema pay
`CREATE TABLE PAY.T1 (COL1 INT)`
- Mel can grant privileges to other users:
 - `GRANT CREATEIN ON SCHEMA PAY TO USER CAL`
 - `GRANT ALTERIN, CREATEIN, DROPIN ON SCHEMA PAY TO GROUP G1 WITH GRANT OPTION`
- Achieving greater schema control:
 - `REVOKE IMPLICIT_SCHEMA ON DATABASE FROM PUBLIC`
 - `GRANT IMPLICIT_SCHEMA ON DATABASE TO USER JON`

© Copyright IBM Corporation 2007

Figure 11-6. Controlling use of schemas

CF238.3

Notes:

A schema can be defined whenever it is desirable to group a set of objects. This grouping may be based on objects for such things as an application, a group of people, or an individual user.

The schema is defined explicitly using the CREATE SCHEMA statement. An option (AUTHORIZATION) on the CREATE SCHEMA statement allows owner specification at the time the schema is created. A schema is defined to have an owner. The owner of the schema is given the privilege to create objects using the schema name as the object qualifier and to drop any objects that are defined in the schema (regardless of definer). The schema owner also has the privilege to grant the privilege to create objects in the schema or the privilege to drop objects from the schema to other users. This means that the schema owner has the ability to manage objects in the schema and may grant similar ability to manage those objects to others.

When a new database is created, PUBLIC is given IMPLICIT_SCHEMA database authority. With this authority, any user can create a schema by creating an object and specifying a schema name that does not already exist. SYSIBM becomes the owner of the

implicitly created schema and PUBLIC is given the privilege to create objects in this schema.

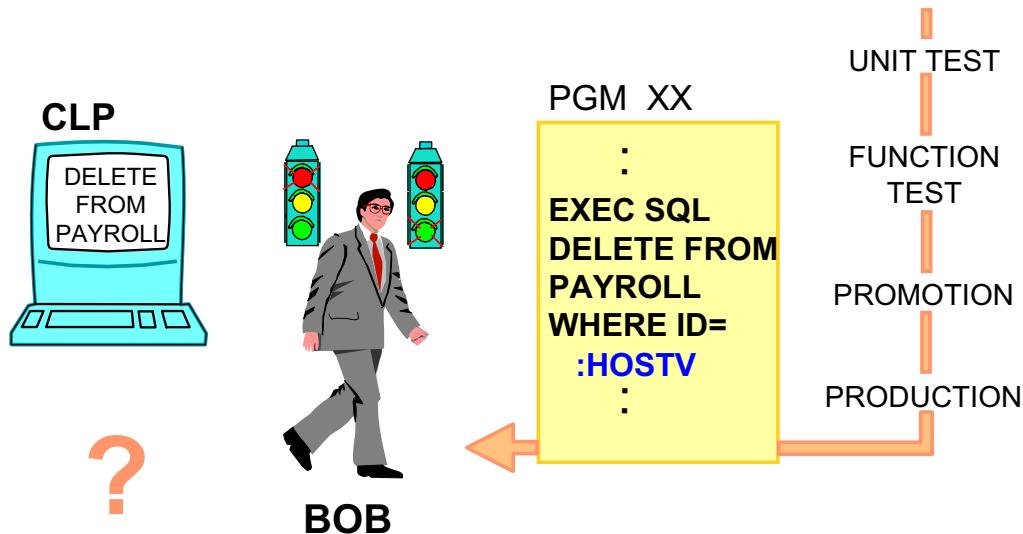
If control of who can implicitly create schema objects is required for the database, IMPLICIT_SCHEMA database authority should be revoked from PUBLIC. Once this is done, there are only three ways that a schema object is created:

- Any user can create a schema using their own authorization name on a CREATE SCHEMA statement.
- Any user with DBADM authority can explicitly create any schema which does not already exist, and can optionally specify another user as the owner of the schema.
- Any user with DBADM authority has IMPLICIT_SCHEMA database authority (independent of PUBLIC) so that they can implicitly create a schema with any name at the time they are creating other database objects.

SYSIBM becomes the owner of the implicitly created schema and PUBLIC has the privilege to create objects in the schema.

A user always has the ability to explicitly create their own schema using their own authorization name.

Protecting resources through programs



© Copyright IBM Corporation 2007

Figure 11-7. Protecting resources through programs

CF238.3

Notes:

An individual can EXECUTE a package without having the authority for the underlying SQL statements.

Programs can be coded, tested, and installed to perform database tasks on behalf of the program executor.

The authority to EXECUTE a program which performs some SQL statement does not imply authority to perform the same SQL in an adhoc environment, such as CLP or Lotus Approach.

11.3 Explicit privileges

Explicit authorization



* GRANT/ REVOKE	Database privileges	ON DATABASE	TO/ FROM	USER/ GROUP	<i>userid groupid PUBLIC</i>
	Package privileges	ON PACKAGE package_name			
	Table/view privileges	ON TABLE table/view_name			
	CONTROL	ON INDEX index_name			
	Schema privileges	ON SCHEMA schema_name			
	USE	OF TABLESPACE tablespacename			

** must be SYSADM, DBADM, or have CONTROL on object*

© Copyright IBM Corporation 2007

Figure 11-8. Explicit authorization

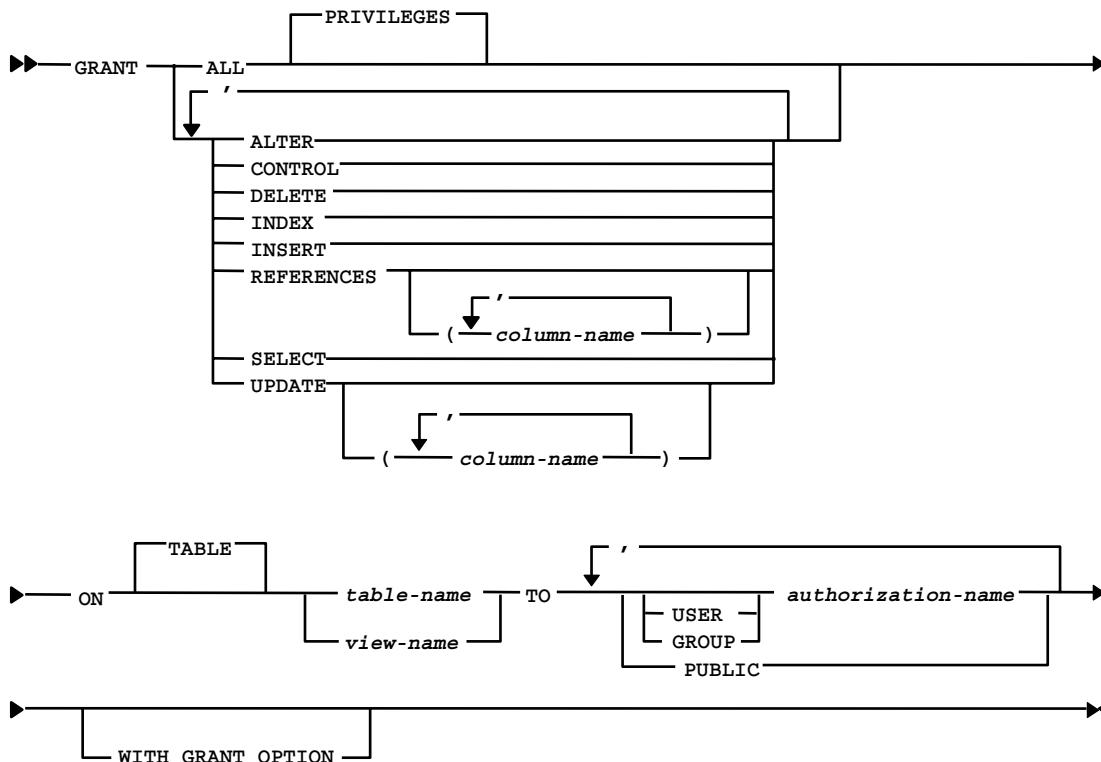
CF238.3

Notes:

The privileges and authorities can be obtained either implicitly or explicitly. You should be aware of all the ways users can obtain access to objects in a database.

Privileges and DBADM authority may be explicitly GRANTed to users or PUBLIC by someone who holds a higher authority or privilege (SYSADM, DBADM, or CONTROL). When granting database privileges, the name of the database is not specified, because you must be CONNECTed (using) a specific database in order to execute SQL statements. Therefore, DB2 knows to which database you are connected.

Grant statement — table / view privileges syntax



© Copyright IBM Corporation 2007

Figure 11-9. Grant statement — table / view privileges syntax

CF238.3

Notes:

The WITH GRANT OPTION is available on the GRANT statement to indicate that the privilege can also be granted by the grantee to other users.

The ability to grant privileges at the column level is available on the GRANT statement for UPDATE (to indicate that the grantee can only update certain columns) or for REFERENCES (to indicate that the grantee can only define a referential constraint for certain columns).

The following example shows the contents of the table authorizations after using the GRANT option in various formats.

USER PATTI IS SYSADM.

```
grant all on pay.tb1 to cal
select * from syscat.tabauth where grantee='CAL'
```

The table below shows that CAL does NOT have CONTROL on TB1. CAL can NOT grant privileges to other users.

Grantor	Grantee	Schema	Tabname	Control	Alter	Delete	Index	Insert	Select	Ref	Update
PATTI	CAL	PAY	TB1	N	Y	Y	Y	Y	Y	Y	Y

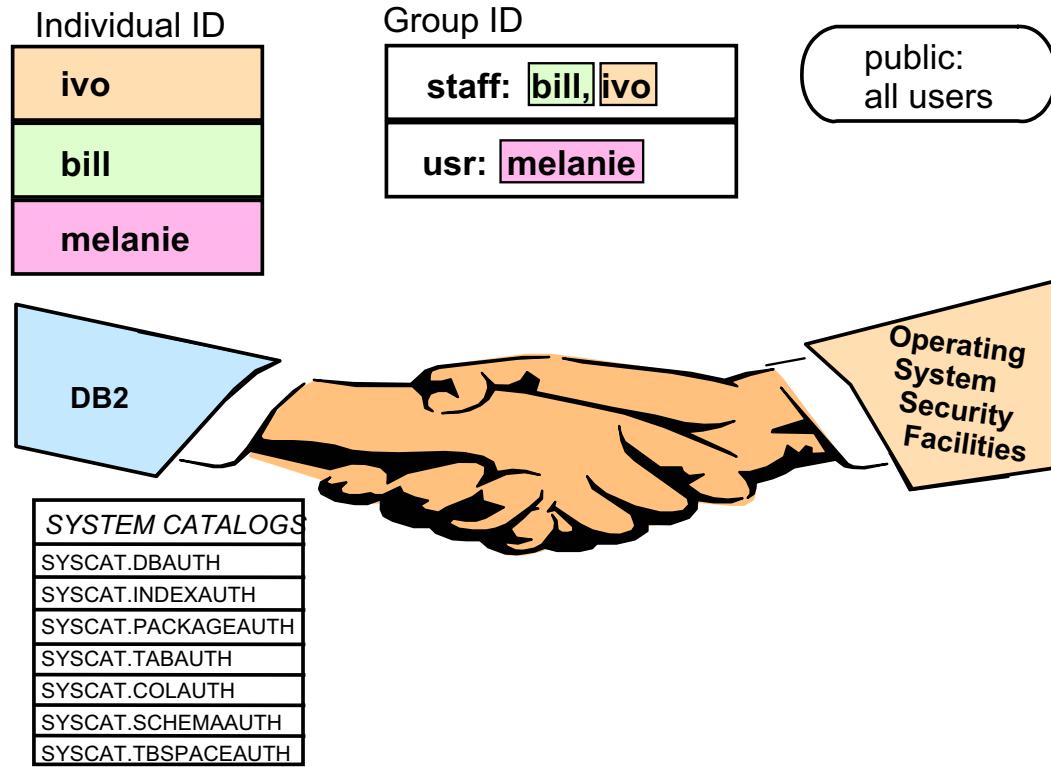
```
grant select,insert on pay.tb2 to JON with grant option
select * from syscat.tabauth where grantee='JON'
```

Grantor	Grantee	Schema	Tabname	Control	Alter	Delete	Index	Insert	Select	Ref	Update
PATTI	JON	PAY	TB2	N	N	N	N	G	G	N	N

```
grant control on pay.tb3 to mel
select * from syscat.tabauth where grantee='MEL'
```

Grantor	Grantee	Schema	Tabname	Control	Alter	Delete	Index	Insert	Select	Ref	Update
PATTI	ME:	PAY	TB3	Y	G	G	G	G	G	G	G

Individual IDs, group IDs, and public



© Copyright IBM Corporation 2007

Figure 11-10. Individual IDs, group IDs, and public

CF238.3

Notes:

Privileges may be explicitly granted to individual IDs, group IDs, or PUBLIC.

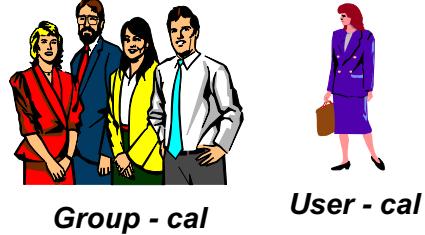
Individual IDs are created through the operating system's security facilities.

A group ID consists of one or more individual IDs. If an individual ID is a member of a group, then that individual ID automatically has privileges granted to the group ID. Groups are created through the operating system's security facilities.

DB2 has a special group ID: PUBLIC. It is not created with the operating system's security facilities. All users accessing a database are automatically members of the group PUBLIC. Granting a privilege to PUBLIC grants that privilege to all users of a database.

Using group IDs

	Permitted on		Does the System Know About?		
	SQL	UNIX	OS/2 or Windows NT	User - cal	Group - cal
1				↙	
2					↙
3			N/A	↙	↙



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

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

3 X GRANT SELECT ON TABLE EMPLOYEE TO CAL → SQLCODE -569

© Copyright IBM Corporation 2007

Figure 11-11. Using group IDs

CF238.3

Notes:

Some operating systems allow the same name to be created for a userid and a group ID. This can cause problems for authorization and privilege checking within DB2. The GRANT and REVOKE SQL statements provide an optional parameter which is used to indicate if the privilege is intended for a userid or a group ID.

Consider the above scenarios:

1. A user called cal exists on the operating system. The GRANT command shown can explicitly use the TO USER clause in the statement. If the TO USER clause is not specified, userid cal will get the SELECT privilege.
2. Only a group called cal exists on the operating system. The GRANT command can explicitly use the TO GROUP clause. If not, because there is no userid cal, the group will receive the privilege.
3. UNIX allows a user and group to have the same name. In UNIX, the following would occur:

- If the GRANT does not specify either TO USER or TO GROUP, the grant will fail with a -569 SQLCODE.
- If the GRANT specifies TO USER or TO GROUP, the exact receiver of the privilege will be specified.

If a privilege has been granted to both a userid and a group ID with the same name, you must specify the GROUP or USER keyword when revoking the privilege. Revoking a privilege from a group may not revoke it from all members of the group. If an individual ID has been directly granted a privilege, it will keep it until that privilege is directly revoked.

Public privilege support for static SQL and views

Mel is a member of group1

```
db2 grant select on table T1 to group group1
db2 grant update on table T1 to user mel
db2 grant insert on table T1 to public
db2 grant select on table T2 to public
```



Mel attempts bind of pgm.sqc

```
exec sql insert into T1...
exec sql update T1 set...
exec sql select... from T1...
```



Mel attempts to create a view

```
create view V2 as select C3 from T2
create view V1 as
    select C1, C8 from T1
```



© Copyright IBM Corporation 2007

Figure 11-12. Public privilege support for static SQL and views

CF238.3

Notes:

Privileges granted to a group will **not** be considered when checking static SQL statements and creating views. However, privileges granted to PUBLIC **will** be considered when checking static SQL statements and creating views.

The graphic illustrates the following points. Mel is a member of group1. When Mel attempts the bind of pgm.sqc:

- The insert to T1 is valid because insert on T1 has been granted to PUBLIC.
- The update of T1 is valid because update on T1 has been granted to user MEL.
- The select statement from T1 is NOT valid because select on T1 has not been granted to user MEL or to PUBLIC.

When Mel attempts the create of view V2:

- The select of column C3 from T2 is valid because select on T2 has been granted to PUBLIC.

When Mel attempts the create of view V1:

- The select of columns C1 and C8 from T1 is NOT valid because select on T1 has not been granted to user MEL or to PUBLIC.

Privileges required for programming



Action	Privileges Required
Precompile to bindfile	CONNECT on database
Create a new package	CONNECT on database BINDADD on database Privileges need to execute each static SQL statement
Modify an existing package	CONNECT on database BIND on package Privileges need to execute each static SQL statement
Re-create an existing package	CONNECT on database BIND on package
Execute a package	CONNECT on database EXECUTE on package
Drop a package	CONNECT on database CONTROL on package or creator of package

© Copyright IBM Corporation 2007

Figure 11-13. Privileges required for programming

CF238.3

Notes:

This chart summarizes the authorities and privileges needed to prepare, bind, modify, execute, or drop a package.

Simple references are qualified with the ID of the binder.

At prepare time, simple references are qualified with the ID of the preparer for the purpose of determining authorization for doing the prepare.

The binder **must** have authorization, granted to their userid or to PUBLIC, to perform all underlying SQL statements at bind time. The preparer is not required to have the underlying SQL authorizations to create a bind file; warnings will be generated if the user ID doing the prepare does not have the authorizations to perform all underlying SQL statements. Also, all objects must exist in the database against which the prepare and the bind are being executed.

Dynamic rules and dynamic SQL



- Jill issues:

```
db2 bind myapp.bnd  qualifier u1  owner u2
    dynamicrules run
```

- Unqualified SQL uses **u1** schema
- u2** owns package
- Privileges of user executing package checked for
- Dynamic SQL


```
db2 bind myapp.bnd qualifier u1  owner u2
          dynamicrules bind
```
- Privileges of **u2** checked for **dynamic SQL**

© Copyright IBM Corporation 2007

Figure 11-14. Dynamic rules and dynamic SQL

CF238.3

Notes:

With the default settings, the owner of a package is the auth ID used to bind the package (in the example, Jill), and that user ID is also used to qualify unqualified references, whether those unqualified references are found in dynamic or static SQL.

Use of the QUALIFIER option allows you to specifically indicate the qualifier to be used for both dynamic and static SQL.

Use of the OWNER option allows you to specifically indicate the user ID to be used for the package owner. Also, the authorization for the static SQL statements in this package is checked against this ID. Only a user with SYSADM or DBADM authority can specify an authorization identifier other than the user ID.

Use of DYNAMICRULES allows you to indicate whether the package user's privileges (the privileges of the user ID executing the package) are checked for dynamic SQL (default, DYNAMICRULES RUN), or whether the implicit or explicit value of the owner of the package should be checked for dynamic SQL (DYNAMICRULES BIND). Note that the value of the CURRENT SCHEMA special register will have no impact on the qualification of SQL in a package bound with DYNAMICRULES BIND.

For dynamic applications:

- To precompile to a bind file, CONNECT on database is required.
- To create a new package, CONNECT on database and BINDADD on database are required.
- To modify an existing package, CONNECT on database and BIND on package are required.
- To recreate an existing package, CONNECT on database and BIND on package are required.
- To execute a package, CONNECT on database and EXECUTE on package are required. In addition, if the package was bound with DYNAMICRULES RUN, privileges needed for executing each dynamic SQL statement are required. If the package was bound with DYNAMICRULES BIND, the binder would require privileges for executing each dynamic SQL statement.
- To drop a package, CONNECT on database and CONTROL on package or creator of package are required.

DB2 BIND GRANT option



RICK has
BINDADD
and
underlying
SQL
privileges

db2 connect to sample
db2 prep packa.sqc bindfile

db2 bind packa.bnd grant

[authid
groupid
public]

OR



SALLY
has
CONTROL
on packa

grant execute, bind on
package packa

to

[authid
groupid
public]

© Copyright IBM Corporation 2007

Figure 11-15. DB2 BIND GRANT option

CF238.3

Notes:

The `db2 bind` command `grant` option requests that package EXECUTE and BIND privilege be granted to an additional userid or PUBLIC.

If the GRANT option is used when binding a package at a local DB2 database, both EXECUTE and BIND privileges are granted. If the GRANT option is used when binding a package at a remote DRDA AS database, only the EXECUTE privilege is granted.

11.4 Implicit privileges

Implicit privileges (1 of 2)

▪ Create database

- Internal GRANT of DBADM authority with BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, LOAD, and QUIESCE_CONNECT privileges to creator (SYSADM or SYSCTRL)
- Internal GRANT of BINDADD, CREATETAB, CONNECT and IMPLICIT_SCHEMA to PUBLIC
- BIND and EXECUTE privilege on each successfully bound utility to PUBLIC
- SELECT on system catalog tables and views to PUBLIC
- USE privilege on USERSPACE1 table space to PUBLIC
- EXECUTE WITH GRANT privilege to PUBLIC on all functions in SYSFUN schema; EXECUTE privilege to PUBLIC on all procedures in SYSIBM schema

© Copyright IBM Corporation 2007

Figure 11-16. Implicit privileges (1 of 2)

CF238.3

Notes:

The privileges and authorities can be obtained either implicitly or explicitly. You should be aware of all the ways users can obtain access to objects in a database.

Some privileges are granted to users automatically (implicit) by DB2. Be aware of these internal GRANTS, so that you can REVOKE the privileges if needed.

When a user creates a database via SYSADM or SYSCTRL authorization, the user is granted DBADM authority for the database. If a user has their SYSADM or SYSCTRL authority removed, you must explicitly revoke DBADM authority for each database that this user created if you wish to prevent them from accessing a database as a DBADM. The creator is granted the privileges that would be given if DBADM had been granted explicitly.

Implicit privileges (2 of 2)



- GRANT DBADM

- Internal GRANT of BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, LOAD and QUIESCE_CONNECT

- Create object (table, index, package)

- Internal GRANT of CONTROL to object creator

- Create view

- Internal GRANT to intersection of creator's privileges on base tables to view creator

© Copyright IBM Corporation 2007

Figure 11-17. Implicit privileges (2 of 2)

CF238.3

Notes:

When a user is granted DBADM authority on a database, the user is also granted BINDADD, CONNECT, CREATETAB, CREATE_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, LOAD, and QUIESCE_CONNECT privileges automatically. **If the DBADM authority is later revoked, the other privileges remain in effect unless they are separately revoked.**

When a user is granted CONTROL privilege on a table, the user is also granted all of the other table privileges automatically. **If CONTROL privilege is later revoked, the other privileges will remain in effect unless they are separately revoked. You may also use the ALL option to do so.**

Privileges granted to the creator of a view will never exceed the base table authority the creator possesses. As an example, consider two users, A and B.

A owns table ATABLE.

B has the SELECT authority on ATABLE.

A creates view AVIEW }

```
        } both views use SELECT * FROM ATABLE  
        } WHERE (some condition) as the view definition  
B creates view BVIEW }
```

User A could SELECT from AVIEW, as well as INSERT, UPDATE, and DELETE, through the view AVIEW. In addition, user A could GRANT authority against AVIEW since user A has CONTROL of the view.

User B could SELECT from BVIEW, but that is all. User B did not have INSERT, UPDATE, and DELETE against the table and did not have the authority to GRANT against the base table. The view BVIEW **DOES NOT** provide privileges beyond those that exist on the base table.

The privileges needed to create a view must be explicitly granted to the user or granted to PUBLIC. Group privileges, except for PUBLIC, are not used when a user's privileges to create a view are checked.

Implicit privileges scenarios

- Scenario 1.

ivo is placed in SYSADM group.
ivo creates database DB1
ivo is removed from SYSADM group.
What privileges does ivo retain?

- Scenario 2.

db2 grant dbadm on database to user mel
db2 revoke dbadm on database from user mel
What privileges does mel retain?

- Scenario 3.

db2 grant control on table t1 to user bill
db2 revoke control on table t1 from user bill
What privileges does bill retain?
Write a command to revoke bill's remaining privileges.

© Copyright IBM Corporation 2007

Figure 11-18. Implicit privileges scenarios

CF238.3

Notes:

Can you solve the scenarios?

Grant / Revoke scenarios



- Scenario 1.

```
db2 create view v2 as select name, pay from t1 where dept=50  
db2 grant select on view v2 to user doug  
db2 grant select on table t1 to user pres  
What is difference between doug's & pres's privileges?
```

- Scenario 2.

```
db2 grant select on table tab1 to user gerhard  
db2 grant select on table tab1 to group denmark  
Who can create a view with tab1 as the base table?  
db2 revoke select on table tab1 from group denmark  
If gerhard in denmark group, may he select from tab1?
```

- Scenario 3.

```
db2 grant select on table tab1 to mel  
Then mel creates view:  
db2 create view v1 as select * from tab1  
db2 revoke select on table tab1 from mel  
What happens to the view v1 and packages that specify v1?
```

© Copyright IBM Corporation 2007

Figure 11-19. Grant / Revoke scenarios

CF238.3

Notes:

Can you answer the above scenarios?

11.5 Querying privileges/authorities

Query who has privileges

- Most of the information on authorizations is maintained in seven system catalog tables:

<i>SYSCAT.DBAUTH</i>	Database privileges
<i>SYSCAT.COLAUTH</i>	Table and View Column privileges
<i>SYSCAT.INDEXAUTH</i>	Index privileges
<i>SYSCAT.PACKAGEAUTH</i>	Access Package privileges
<i>SYSCAT.SCHEMAAUTH</i>	Schema privileges
<i>SYSCAT.TABAUTH</i>	Table and view privileges
<i>SYSCAT.TBSPACEAUTH</i>	Table space privileges

- SYSADM, SYSCTRL, SYSMAINT, and SYSMON authority and group membership are defined outside the Database Manager and not reflected in catalogs

© Copyright IBM Corporation 2007

Figure 11-20. Query who has privileges

CF238.3

Notes:

During database creation, SELECT privilege on the system catalog views is granted to PUBLIC. In most cases, this does not present any security problems. For very sensitive data, it may be inappropriate, as these tables describe every object in the database. If this is the case, consider revoking the SELECT privilege from PUBLIC; then grant the SELECT privilege as required to specific users or groups. Granting and revoking SELECT on the system catalog views is done in the same way as for any view, but you must have either SYSADM or DBADM authority to do so.

At a minimum, you should consider restricting access to the SYSCAT.DBAUTH, SYSCAT.COLAUTH, SYSCAT.INDEXAUTH, SYSCAT.PACKAGEAUTH, SYSCAT.SCHEMAAUTH, SYSCAT.TABAUTH, and SYSCAT.TBSPACEAUTH catalog views. This would prevent information on user privileges, which could be used to target an ID for break-in from becoming available to everyone with access to the database.

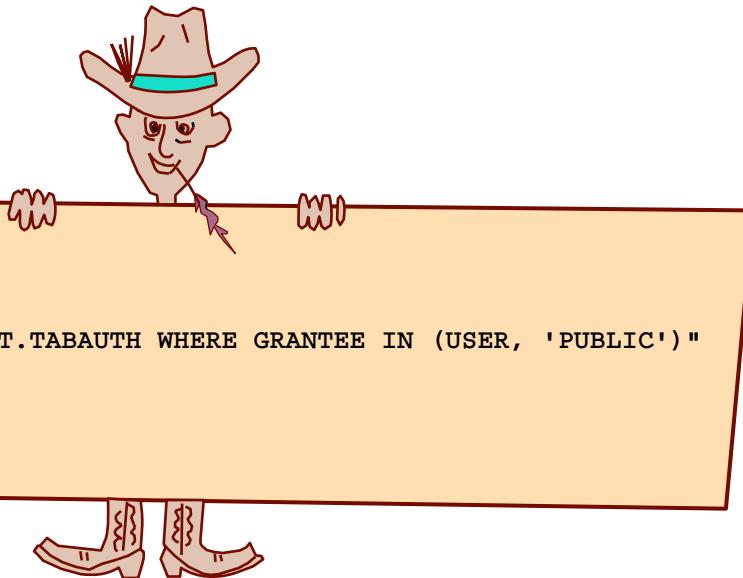
You should examine the columns for which statistics are gathered. Some of the statistics recorded in the system catalog contain data values which could be sensitive information in your environment. If these statistics contain sensitive data, you may wish to revoke

SELECT privilege from PUBLIC for the SYSCAT.COLUMNS and SYSCAT.COLDIST catalog views.

If you wish to limit access to the system catalog views, you could define views on the views to let each authorization ID retrieve information about its own privileges. This may be done by using a *WHERE IN (USER, 'PUBLIC')* predicate.

Privileges granted to users by the system will have SYSIBM as the grantor. SYSADM, SYSMAINT, SYSCTRL and SYSMON authorizations are not listed in the system catalog.

Query privileges granted to current ID



```
db2 "SELECT * FROM SYSCAT.TABAUTH WHERE GRANTEE IN (USER, 'PUBLIC')"
```

Rounding up Privileges

© Copyright IBM Corporation 2007

Figure 11-21. Query privileges granted to current ID

CF238.3

Notes:

USER is a keyword that means the current *connection ID*. Do not place USER in quotation marks.

If you had a current operating system login of USER1 and then connected to database DB1 with “CONNECT TO DB1 USER MELANIE USING SCUBA”, USER would equal MELANIE.

DB2 get authorizations reports the authority and privileges that you have for the database. You must be connected to the database to get authorizations.

Direct authority or privileges are acquired by explicit commands that grant the authority or privilege to your username. Indirect authority or privileges are through implicit ways; that is, SYSADM has authority over the entire DB2 system.

11.6 Label Based Access Control (LBAC)

Label Based Access Control (LBAC) overview

- Label-Based Access Control (LBAC) is an implementation of Mandatory Access Control (MAC)
 - In the context of database systems, LBAC is commonly referred to as the ability to control access to data rows based on security labels.
- The DB2 LBAC solution is a *flexible* implementation of MAC at both:
 - The Row level
 - The Column level
- Row level and column level protection are independent and can be used either separately or combined
- Table level labeling can be easily simulated using column level labeling
- LBAC complements the traditional DB2 Discretionary Access Control (DAC)
 - When a protected table is accessed, DB2 enforces two (2) levels of access control:
 1. DAC at the table level, that is, does the user hold the required privilege to perform the requested operation on the table?
 2. LBAC, which can be at the row level, column level or both

© Copyright IBM Corporation 2007

Figure 11-22. Label Based Access Control (LBAC) overview

CF238.3

Notes:

Label-Based Access Control (LBAC) is an implementation of Mandatory Access Control (MAC). In the context of database systems, LBAC is commonly referred to as the ability to control access to data rows based on security labels.

The DB2 LBAC solution is a flexible implementation of MAC that can provide row level and column level protection. Row level and column level protection are independent and can be used either separately or combined. Table level labeling can be easily simulated using column level labeling for each column in a table.

LBAC complements the traditional DB2 Discretionary Access Control (DAC). For access to tables protected using LBAC functions, DB2 enforces two levels of access control:

- First, the standard Discretionary Access Control permissions are checked. DB2 checks the authorities that have been granted to a user or user groups to see if the access is permitted. For example, does the user have SELECT authority for a table or view?
- Next, DB2 would check for LBAC authorization to access protected rows or columns in a table.

Security model extensions — SECADM Authority



- Security Administrator (SECADM) Authority
 - The authority required to
 - Create and drop security label components
 - Create and drop security policies
 - Create and drop security labels
 - Grant and revoke security labels
 - Grant and revoke exemptions
 - Grant and revoke the SETSESSIONUSER privilege
 - SYSADM is the only authority that can grant SECADM
 - SECADM has no inherent ability to access data within a protected table!
 - SECADM cannot grant any privileges to him/her self!
 - SYSADM authority also does not provide access to data that is protected by LBAC.

© Copyright IBM Corporation 2007

Figure 11-23. Security model extensions — SECADM Authority

CF238.3

Notes:

Implementing Label Based Access Control security requires the use of a new user authority, the Security administration authority, or SECADM. In order to support this new higher level of security, it is necessary to establish one or more users who must perform all maintenance for the new LBAC objects.

SECADM authority can only be granted by the SYSADM user and can be granted to a user but not to a group. It gives these and only these abilities:

- Create and drop security label components
- Create and drop security policies
- Create and drop security labels
- Grant and revoke security labels
- Grant and revoke LBAC rule exemptions
- Grant and revoke setsessionuser privileges
- Execute the SQL statement TRANSFER OWNERSHIP on objects that you do not own

No other authority gives these abilities, not even SYSADM.

In DB2 databases, the SYSADM user has had *unlimited* permission to perform any command or data access. The abilities provided by the SECADM authority are not provided by any other authority. SYSADM authority also does not provide access to data that is protected by LBAC.

The SECADM user has no inherent ability to access data within a protected table, and a SECADM user cannot grant any privileges to him/her self. So the security administrator manages the new LBAC security objects for other users, but does not have any special authority to access data.

Checkpoint

Exercise — Unit Checkpoint

1. Can the right to grant table, view, or schema privileges to others be extended to other users using the WITH GRANT OPTION on the GRANT statement?
-

Unit summary

Having completed this unit, you should be able to:

- Differentiate between authentication and authorization
- Use DB2 access control mechanisms to implement security within the database
- Use group ids to create a control hierarchy
- Describe Label Based Access Control (LBAC)
- Explore privileges in the database
- Describe privileges required for binding and executing a package
- Describe the difference between explicit privileges and implicit privileges
- Describe the different DB2 authorization levels

© Copyright IBM Corporation 2007

Figure 11-24. Unit summary

CF238.3

Notes:

Appendix A. LOBs, UDFs, UDTs, and Relational Extenders

What This Unit Is About

This unit provides information about LOBs, UDFs, UDTs, and Relational Extenders.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe LOBs, UDFs, UDTs, and Relational Extenders

References

Command Reference

Administration Guide

How You Will Check Your Progress

Accountability:

- Machine lab
- Checkpoint Questions

Appendix objectives



After completing this appendix, you should be able to:

- Describe LOBs, UDFs, UDTs, and Relational Extenders

© Copyright IBM Corporation 2007

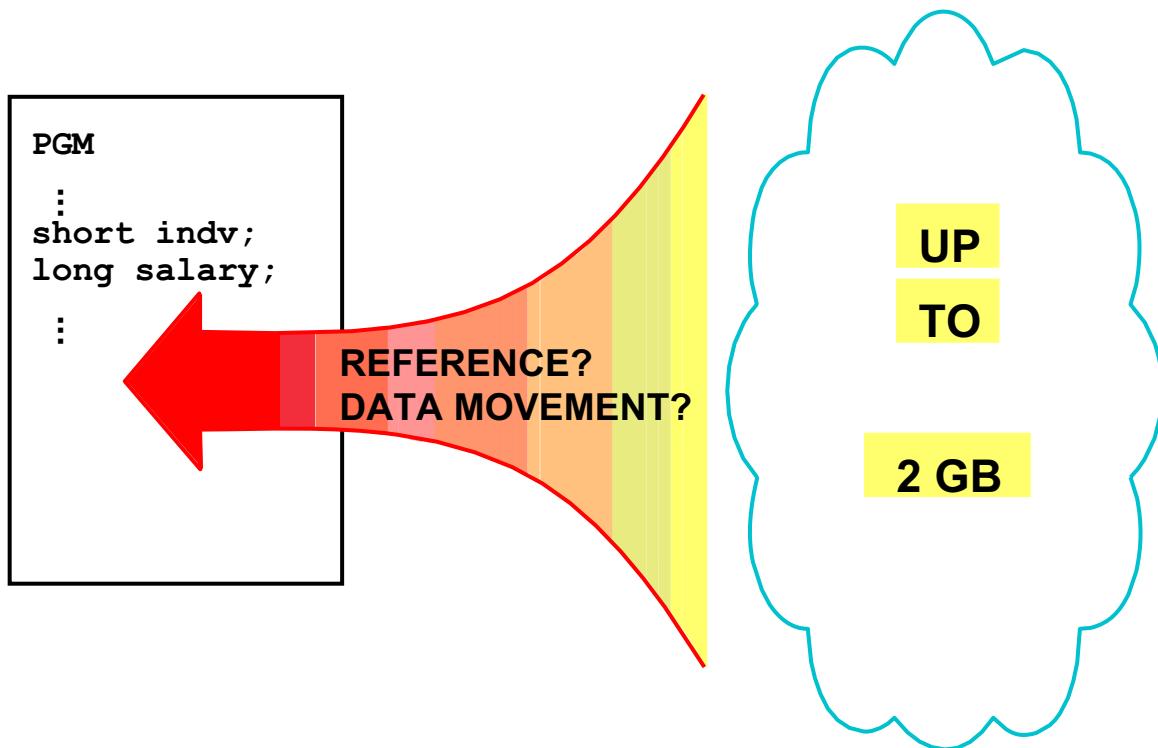
Figure A-1. Appendix objectives

CF488.3

Notes:

A.1. LOBs

Large objects — memory consideration



© Copyright IBM Corporation 2007

Figure A-2. Large objects — memory consideration

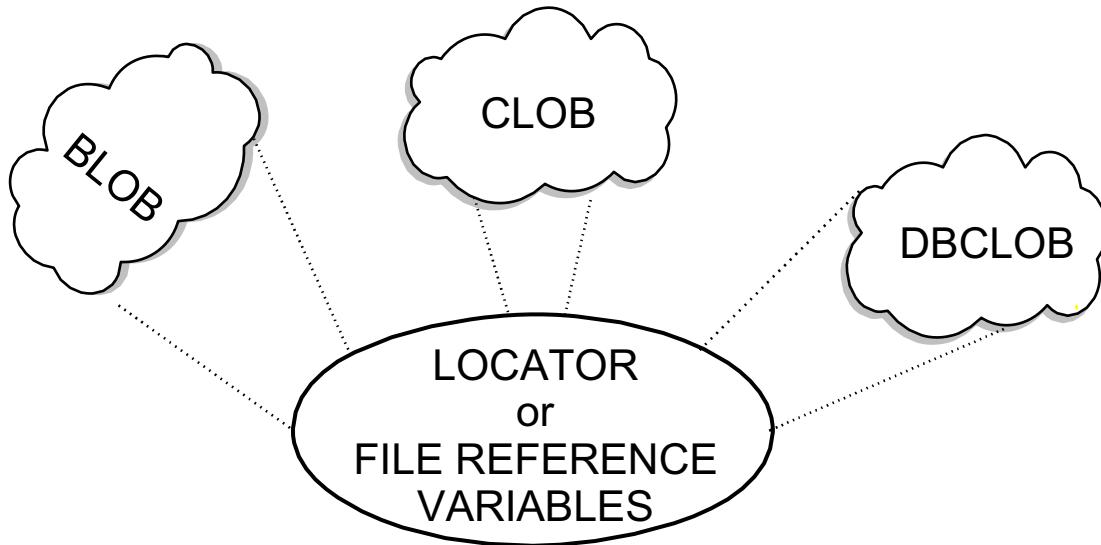
CF488.3

Notes:

Storage of LOB data is now possible due to improvements in storage systems and technology.

The programmer should still be concerned with data movement of such large data volumes. For example, moving two gigabytes into an application when the entire LOB is not required would be wasteful in terms of resource consumption. In addition, using application memory for such a large object is not feasible.

Large objects — manipulation

- ✓ Reduce data transfer
- ✓ Defer evaluation for performance
- ✓ Direct file transfer
- ✓ CLI Functions

© Copyright IBM Corporation 2007

Figure A-3. Large objects — manipulation

CF488.3

Notes:

DB2 provides the programmer with mechanisms to manipulate LOB data so that performance impact can be minimized.

A Locator Variable can be defined to represent the LOB or a portion of LOB data. A Locator Variable acts like a token within the application. The use of tokens allows the programmer to retrieve the LOB in parts, thus reducing the amount of data actually transferred to an application.

DB2 will permit several operations on LOB data:

- Retrieve value or partial value
- Replace value
- Use of the LIKE predicate
- Use of concatenation
- Use of the scalar functions SUBSTR, POSSTR, and LENGTH
- Use of UNION ALL

Evaluation of LOB data predicates is considered during the optimization process and will generally be deferred to enhance performance. The LOB data will not be accessed until other predicates are applied to reduce the number of candidate rows.

File reference variables provide the application programmer with a method to move LOB data directly between files and the database. There is no need to move the data into program storage when using file reference variables.

DB2 Relational Extenders



- Help DB2 users
 - Handle complex data types
 - Improve application development productivity
 - Reduce development complexity
- Text Extender
- Image Extender
- Video Extender
- Audio Extender
- XML Extender
- Extra Charge:
 - Net Search Extender
 - Spatial Extender

© Copyright IBM Corporation 2007

Figure A-4. DB2 Relational Extenders

CF488.3

Notes:

Text Extender components consist of:

- Administration CLP for preparing databases, tables, and columns for text search
- Building text indexes and changing text index characteristics
- Setting and changing default document characteristics such as language, format, and codepage
- Text extender UDFs for integrating full-text search into SQL queries, retrieving document characteristics, and refining text search arguments
- Programming API (C) for performing text search, browsing the resulting documents, including highlighting of matches

Image Extenders can be used to:

- Store and query images like traditional data
- Support a variety of image formats like GIF, JPEG, BMP, and TIFF
- Import and export images with attributes like size in bytes, format, height, width, and number of colors
- Convert the format of the images

- Query images based on related data or image attributes
- Generate and display thumbnail images
- Query By Image Content (QBIC) technology to search by what you see, match color, match texture, and match shape

Video Extenders can be used to:

- Store and query video clips like traditional data
- Import and export video clips with attributes like frame rate, compression format, and number of video tracks
- Query video clips based on related data or video attributes like name, number, description, format of video, and last update
- Play video clips
- Create an SQL statement within an application to query news databases for video clips
- Select a subject, list the playing times of selected clips, and allow application users to play clips
- Automatically segment a video clip into shots based on scene change detection when the recording camera changes its point of view

Audio Extenders can be used to:

- Store and query audio clips like traditional data
- Import and export audio clips with attributes like audio channels, transfer time, and sampling rate
- Query audio clips based on related data or audio attributes like name, number, description, format of the audio, and last update
- Play audio clips

With the XML Extender, your application can:

- Store entire XML documents as column data in an application table or externally as a local file, while extracting desired XML element or attribute values into side tables for search. Using the XML column method, you can:
 - Perform fast search on XML elements or attributes of SQL general data types that have been extracted into side tables and indexed
 - Update the content of an XML element or the value of an XML attribute
 - Extract XML elements or attributes dynamically using SQL queries
 - Validate XML documents during insertion and update
 - Perform structural-text search with the Text Extender
- Compose or decompose contents of XML documents with one or more relational tables, using the XML collection storage and access method.

Net Search Extender:

- Contains a DB2 stored procedure that adds the power of fast full-text retrieval to Net.Data, Java, or CLI applications.
- Offers application programmers a variety of search functions such as fuzzy search, stemming, Boolean operators, and section search.

Spatial Extender can be used to:

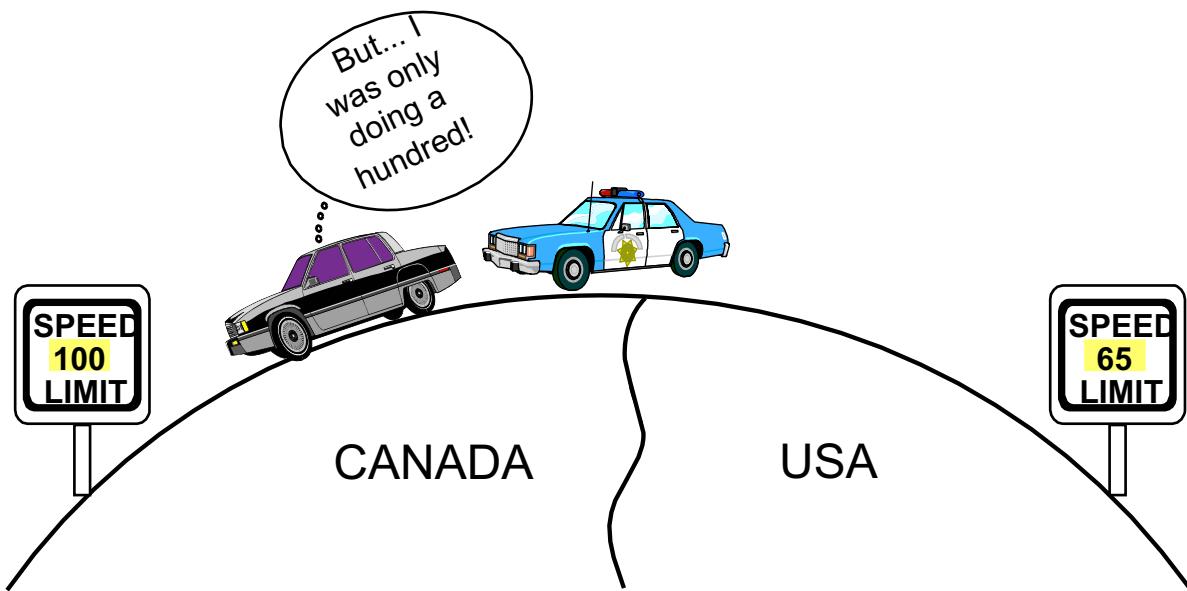
- Create a geographic information system: a complex of objects, data, and applications that allows you to generate and analyze spatial information about geographic features.
- Integrate spatial data with normal SQL data to give users access to new types of queries that could not previously be run.
- Store and index spatial data (coordinate information) and allow spatial queries to run against this data.

A.2. User-Defined distinct types and User-Defined functions

User-Defined Types — the need



- Need to establish context for values
- DB2 enforced typing



© Copyright IBM Corporation 2007

Figure A-5. User-Defined Types — the need

CF488.3

Notes:

In the business world, typing is not limited to data types such as INTEGER or CHARACTER. Installations require data typing that specifically reflects the use of data.

For example, the above diagram represents a situation where the statement "**the speed limit is 100**" is ambiguous until the unit of measure is applied. 100 miles per hour is NOT the same as 100 kilometers per hour.

By permitting the installation to define types to the database manager, code to enforce typing is not required in applications. The database manager will enforce strong typing so that invalid comparisons are not allowed. These types are known as User-Defined Types (**UDTs**).

Strong typing is also valuable for adhoc access to data. The user taking advantage of a query tool may attempt to compare such things as *miles per hour* and *kilometers per hour* without realizing his or her error. The database manager can prevent such comparisons.

User-Defined Types — definition

```

CREATE DISTINCT TYPE KPH AS INTEGER
  WITH COMPARISONS
CREATE DISTINCT TYPE MPH AS INTEGER
  WITH COMPARISONS

CREATE TABLE SPEED_LIMITS
  (ROUTE_NUM  SMALLINT,
   CANADA_SL  KPH NOT NULL,
   US_SL      MPH NOT NULL )

```

```

SELECT ROUTE_NUM FROM SPEED_LIMITS
  WHERE CANADA_SL > KPH(80)

```

```

SELECT ROUTE_NUM FROM SPEED_LIMITS
  WHERE CANADA_SL > US_SL

```

FAILS

© Copyright IBM Corporation 2007

Figure A-6. User-Defined Types — definition

CF488.3

Notes:

IMPLICIT_SCHEMA authority or CREATEIN privilege is required to create a distinct type using a schema name that matches the current authorization ID of the statement. To create a distinct type in another schema, SYSADM or DBADM or CREATEIN privilege is needed.

The CREATE DISTINCT TYPE statement allows installations to create any number of data types that will be handled by the database manager.

The TYPE defined is **SOURCED** on an existing type, that is, one defined with the base product (such as INTEGER).

Once a type is defined, column definitions can reference the type when tables are created or altered. The database manager will enforce strong typing when such columns are used. For example, the SELECT statement shown at the bottom of the visual is invalid since KPH and MPH have been defined as DISTINCT types. They cannot be compared.

The use of the clause **WITH COMPARISONS** causes DB2 to create casting functions to convert the UDT to and from its source type. It is required syntax except when the base type is a LONG VARGRAPHIC, LONG VARCHAR, BLOB, CLOB, or DBCLOB. For

example, the definition of KPH in the illustration creates the function to cast that distinct type to and from the INTEGER source type. The use of **WITH COMPARISONS** and the automatic generation of the casting functions allows the following statement to be valid:

```
SELECT ROUTE_NUM FROM SPEED_LIMITS  
WHERE CANADA_SL > KPH(80)
```

where KPH(80) will cast the integer value of 80 to a type of KPH.

In DB2, structured types may also be created with a CREATE TYPE statement. A user-defined structured type may include zero or more attributes. A structured type may be a subtype allowing attributes to be inherited from a supertype.

Create Distinct Type — GUI

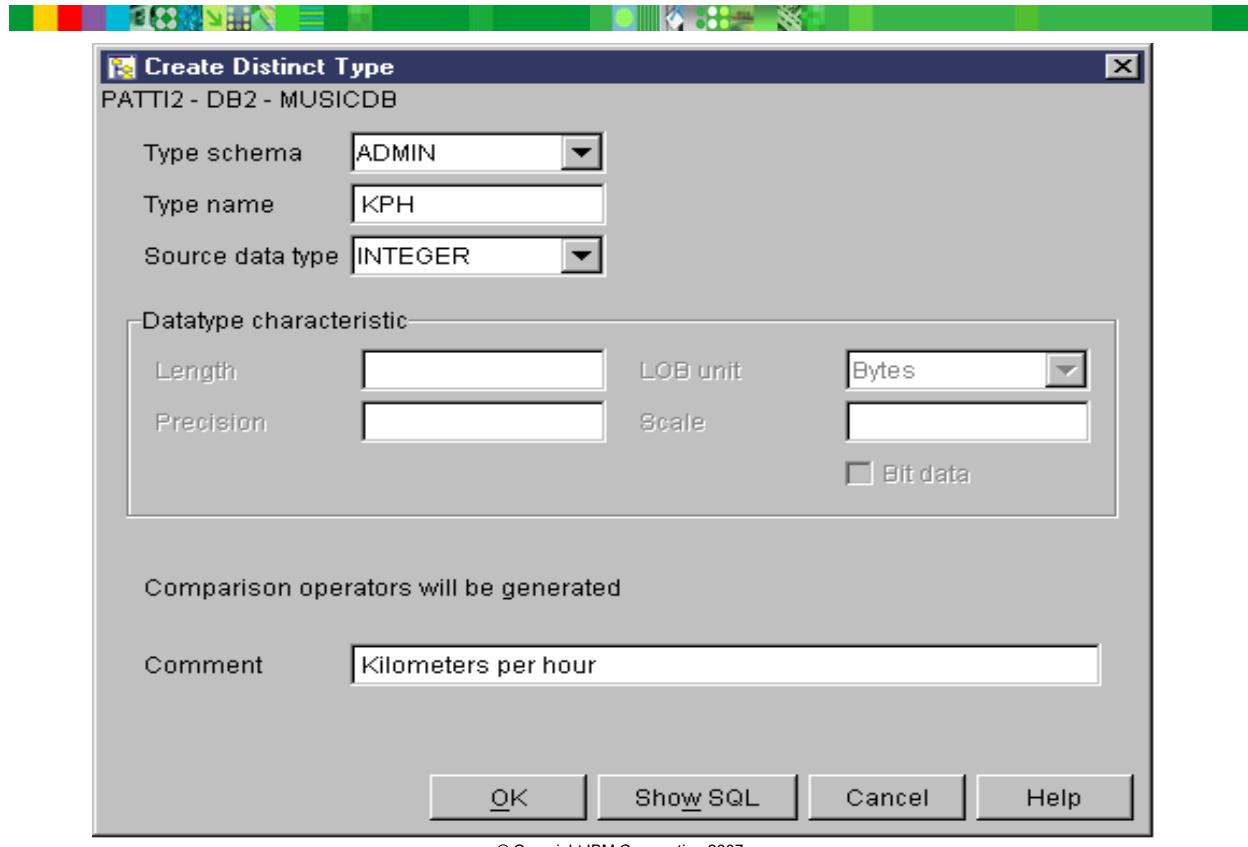


Figure A-7. Create Distinct Type — GUI

CF488.3

Notes:

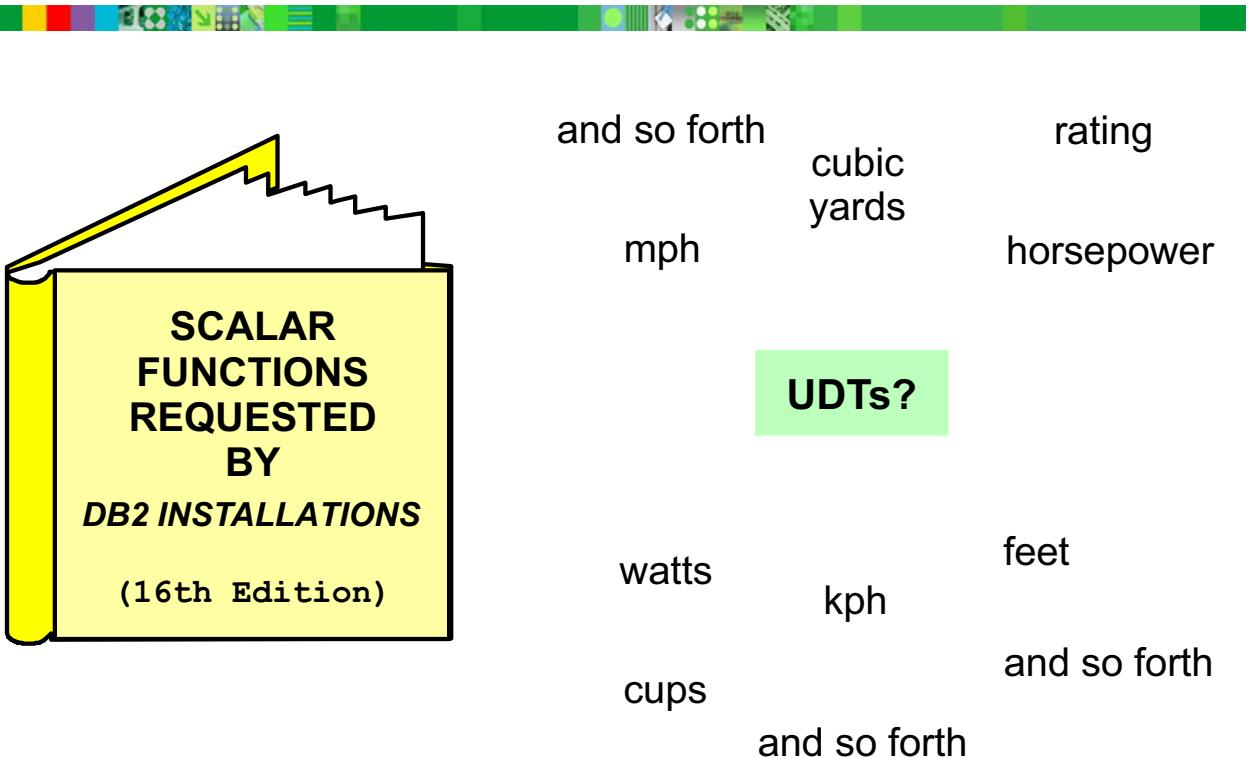
To access the Create Distinct Type GUI from the Control Center:

- Application Objects
 - Select User-Defined Distinct Types (right mouse button)
 - Select Create

The Create Distinct Type GUI allows the following to be indicated:

- Selection for the schema for the distinct type
- Specification of the name of the distinct type
- Select of the source data type
- Datatype characteristic for non-fixed source data types
- Comment

User-Defined Functions — the need



© Copyright IBM Corporation 2007

Figure A-8. User-Defined Functions — the need

CF488.3

Notes:

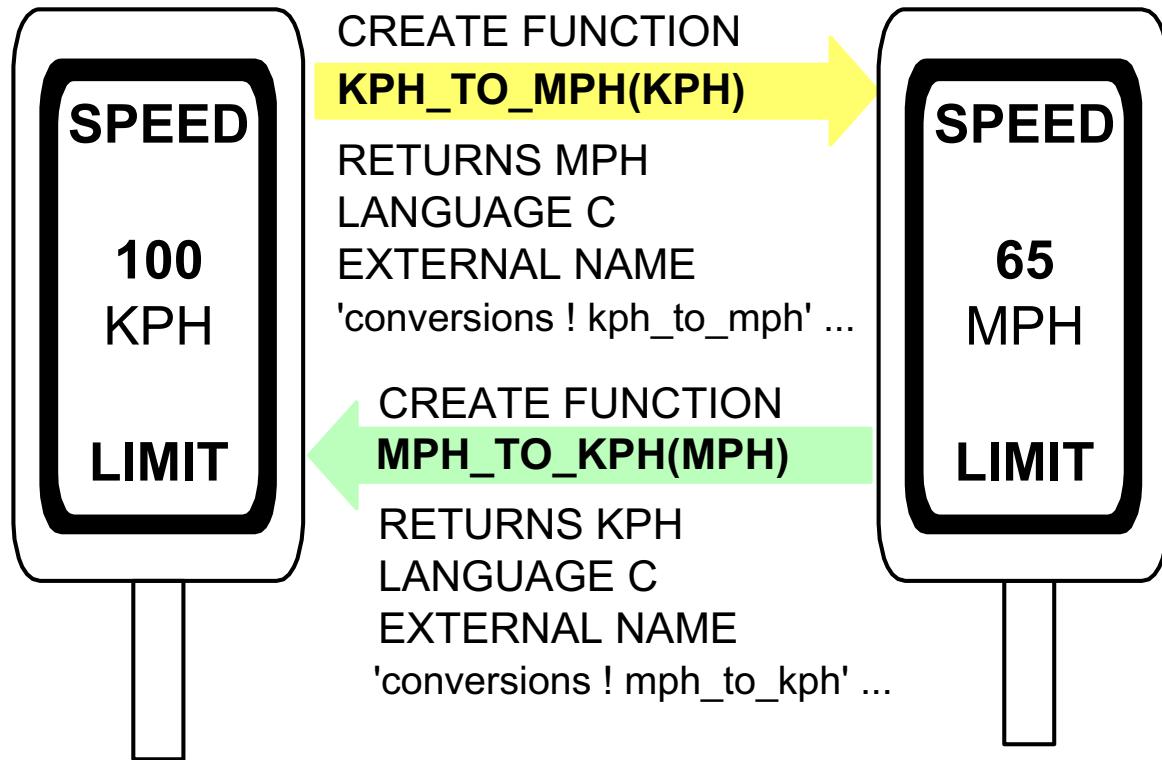
The number of different business applications using DB2 as a database manager is wide and varied, and the number continues to grow. This diverse usage places demands on the database to support a wide variety of scalar functions.

The use of UDTs also increases the need for additional scalar functions. For example, a conversion may be required to convert from gallons to liters.

The introduction of support for User-Defined Functions (**UDFs**) allows the installation to define any number of site specific functions.

This support can also be exploited by software developers who write specialized applications that can be installed with DB2 as the database manager supporting data storage and retrieval. The software developer may provide UDTs and UDFs as part of the software package.

User-Defined Functions — definition



© Copyright IBM Corporation 2007

Figure A-9. User-Defined Function — definition

CF488.3

Notes:

The CREATE FUNCTION statement can be used to define a user-defined SQL scalar, table, or row function. A scalar function returns a single value each time it is invoked, and is generally valid wherever an SQL expression is valid. A table function may be used in a FROM clause and returns a table. A row function may be used as a transform function and returns a row.

The UDF is coded in a programming language such as C, Java or SQL PL. This allows the installation to take advantage of functions available in the language. The installation can take advantage of library management of such functions as well. For example, the functions defined in the visual are conversion functions. They reside in a library called **conversions**.

A UDF can be invoked anywhere in an SQL statement where an expression is currently permitted.

UDFs are considered by the optimizer during access path selection.

For an example of usage of the UDF defined as MPH_TO_KPH, consider resolution of the previously illustrated SELECT statement that needed to compare KPH to MPH.

```
SELECT ROUTE_NUM FROM SPEED_LIMITS
      WHERE CANADA_SL > MPH_TO_KPH(US_SL)
```

A CREATE FUNCTION statement can also be used to create a user defined scalar SQL function. If the returned value of a function can be determined via SQL PL statements, then an SQL scalar function could be used. However, if you need to access resources outside of DB2 (for example, read a file) which SQL PL does not support, then the UDF must be coded in an external language. For example, the function illustrated on the graphic could be coded as:

```
CREATE FUNCTION MPH_TO_KPH (MILES MPH)
  RETURNS KPH
  LANGUAGE SQL
  CONTAINS SQL
  RETURN KPH(INTEGER(1.6 * (1.0 * MILES)))
```

which would require no external code to be created.

Scalar function:

```
CREATE FUNCTION tan (x double) RETURNS double LANGUAGE SQL
  CONTAINS SQL NO EXTERNAL ACTION DETERMINISTIC
  RETURN sin(x) / cos(x)
```

Row function:

```
CREATE FUNCTION fromperson (p person)
  RETURNS ROW (name varchar(10), firstname varchar(10))
  LANGUAGE SQL CONTAINS SQL NO EXTERNAL ACTION DETERMINISTIC
  RETURN VALUES (p..name, p..firstname)
```

Table function:

```
CREATE FUNCTION deptemployees (deptno char(3))
  RETURNS TABLE (empno char(6), lastname varchar(15),
  firstname varchar(12))
  LANGUAGE SQL READS SQL DATA NO EXTERNAL ACTION DETERMINISTIC
  RETURN
    SELECT EMPNO, LASTNAME, FIRSTNAME FROM EMPLOYEE
    WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

User-Defined Functions — sourced

→ CREATE FUNCTION

"+" (MPH, MPH)

RETURNS MPH

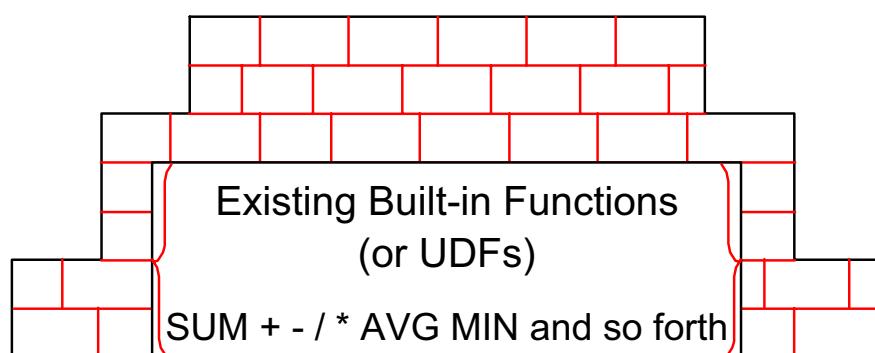
SOURCE "+" (INTEGER, INTEGER)

→ CREATE FUNCTION

AVG_US_SPEED (MPH)

RETURNS MPH

SOURCE AVG(INTEGER)



© Copyright IBM Corporation 2007

Figure A-10. User-Defined Function — sourced

CF488.3

Notes:

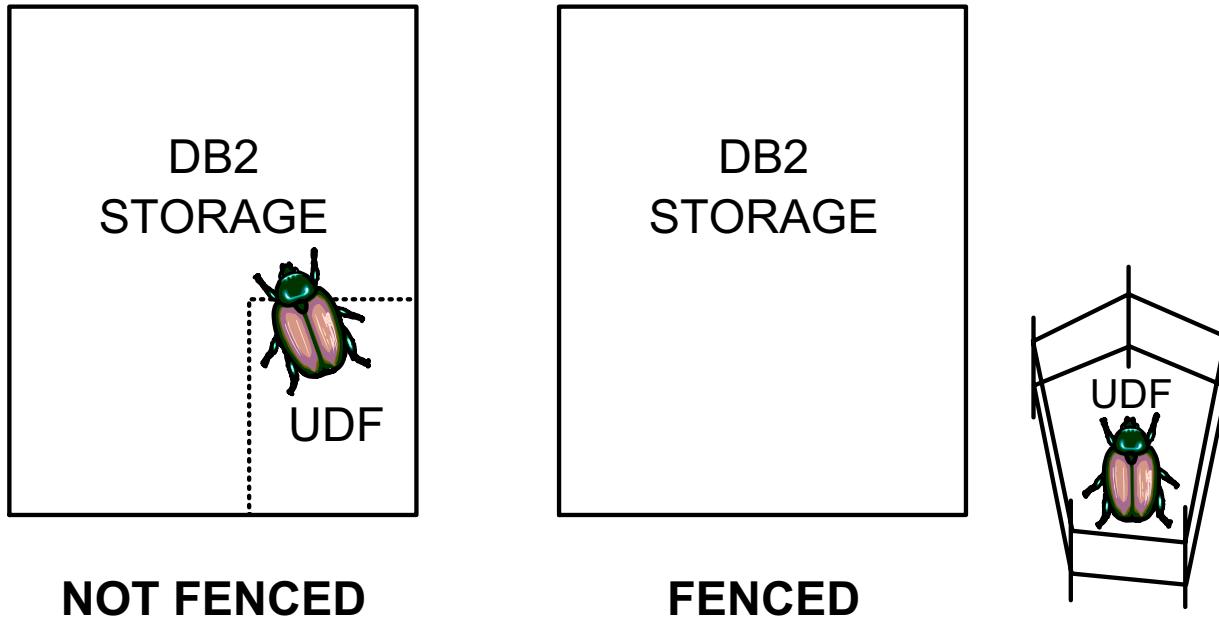
Strong typing defines that data types can only be used in manners defined by the installation. Therefore, a UDT cannot be used in conjunction with functions provided by the database manager unless such usage is defined. To assist the installation in defining common functions, DB2 allows functions to be **sourced** on previously defined functions. These previously defined functions could be provided by the database manager or user-written.

Note: It may be advantageous **NOT** to define a function sourced on every function provided by the database manager for certain UDTs. For example, assume the employee number is defined as a distinct type based on the INTEGER type. The installation would probably not define the functions of SUM, AVG, +, -, *, and / on this data type since such manipulations would be invalid.

Overloading of a function name is also permitted so that the same name can be used several times. For example, consider the function **AVG_US_SPEED** defined in the visual. This function could have been defined as **AVG**, even though such a function name already

exists. The database manager will invoke the function that matches the input parameters passed in an overloading situation.

User-Defined Functions — fencing



© Copyright IBM Corporation 2007

Figure A-11. User-Defined Functions — fencing

CF488.3

Notes:

DB2 provides the installation with the capability to determine where user-defined functions execute. Where UDFs execute is set at the instance level and thus applies to all databases created within an instance.

A function that is *fenced* cannot impair the integrity of the database manager's storage area. If coding errors are present in the UDF, the database will be shielded from the errors.

A function that is not fenced executes as part of the database manager code. This may yield a performance improvement, but care must be taken to ensure that the UDF does not corrupt the database manager storage. The capability to create a function that is not fenced is controlled through DB2 authorization.

On UNIX systems, fenced UDFs can be run under a USERNAME different from the instance USERNAME. The UDF USERNAME must be specified on the DB2ICRT command when an instance is created.

Other clauses that may be defined when creating UDFs include:

- DETERMINISTIC or NOT DETERMINISTIC - function will or will not produce different results when called with the same arguments. For example, a randomizing function would be defined as NOT DETERMINISTIC.
- EXTERNAL ACTION - defines to the database manager that an action external to the database may be taken by the function. For example, a function that sends a message to a development team has an EXTERNAL ACTION.

Checkpoint

Exercise — Appendix A Checkpoint

- ___ 1. What are the data types for large objects?

Appendix summary



Having completed this appendix, you should be able to:

- Describe LOBs, UDFs, UDTs, and Relational Extenders

© Copyright IBM Corporation 2007

Figure A-12. Appendix summary

CF488.3

Notes:

Appendix B. Checkpoint solutions

Unit 4 Checkpoint

1. Name the two types of table spaces:

SMS

DMS

2. Which type of table space must be used if you want to store the index data separately from the table data?

DMS

3. Which type of table space should be used to allow the operating system to allocate pages to the table space as needed?

SMS

Unit 7 Checkpoint

1. State the three methods of recovery.

Crash, version, roll forward recovery

2. What is required to avoid the need for a roll forward?

Offline database backup

Restore...without Rolling Forward

3. How might a database be placed in roll forward pending status?

RESTORE online database backup

RESTORE offline database backup omitting WITHOUT ROLLING FORWARD

4. How might a table space be placed in roll forward pending status?

RESTORE a table space backup. DB2 detects a media failure isolated to a table space.

- ___ 5. What is the minimum time to which a database must be rolled forward?
-

To the end of the time of the online backup.

a table space?

To the minimum roll forward time queried via LIST TABLESPACES.

Unit 8 Checkpoint

- ___ 1. You are evaluating the locking strategy of a program that reads and changes data. The program reads or changes a limited number of rows in the target tables and should run during the time when heavy use of the system is expected by other transactions. What lock modes best support this level of required concurrency at the table level?
-

IS for reading and IX for changing.

Ref: This question is meant to emphasize the most common table locking strategy expected for table spaces and tables in transaction processing.

- ___ 2. A programmer requests your advice. An application that will run during a batch cycle (no other concurrent requests against the target table) will be written to update a large portion of the table. Explain indicates an IX lock for the table space and table. Would you suggest any technique for the programmer to investigate, and if so, what is the technique?
-

Include: LOCK TABLE name IN EXCLUSIVE MODE.

Ref: Eliminating row locking may be beneficial in terms of performance for the application in question. However, the impact on concurrency needs to be carefully evaluated before using the LOCK TABLE statement.

- ___ 3. An airline requires an application that will determine available seats on a particular flight to support the reservation system. The application should list the seats available that meet a general grouping (for example, aisle seat toward the front of the plane) so that reservations can suggest a seat when booking a passenger. However, the application should not prevent update to the row concerning the seat while the passenger is considering possible options. (In other words, the seat is not held for the passenger while the passenger is thinking; in fact, the cursor is closed after producing the answer set.) Assume that the airline uses conversational

transactions (that is, the transaction does not commit prior to displaying a screen). What isolation level would be most suitable?

Cursor Stability, CS.

Ref: The isolation levels of Repeatable Read or Read Stability are not appropriate because they would maintain S locks on rows accessed. This is in conflict with the business requirement for other applications to be able to update the rows examined.

Uncommitted read is not appropriate, since rows in the process of being updated should not be considered *available*. (It is possible that some students may disagree with this interpretation of the term *available*. As long as a student demonstrates understanding of the difference between UR and CS, the interpretation of the term is not a concern.)

Cursor Stability permits obtaining a list of seats that are *available* at the time they were examined, but does not prevent changes to the status when the rows are presented to the reservation agent.

-
- 4. Assume that the airline described in the previous question now wants to ensure that any seat being shown to the user will be available, should that user wish to reserve it. What isolation level would be most suitable?

Read Stability.

Ref: Some people may suggest repeatable read, but this is not appropriate as it will maintain S locks on all rows accessed, not just the ones being presented to the user. Cursor Stability will not lock all of the rows that have been read; in fact, since the cursor was previously closed, none of the rows will be held.

-
- 5. You are the database administrator for a large financial institution. Most processing requirements of the business are accomplished by using an isolation level of Cursor Stability. However, assume that government regulations dictate that a status report of an account table must be generated at 12:00 p.m. on the last day of business each month. Furthermore, the regulations stipulate that the state of all rows is not permitted to change from the start of the report application until it is completed. How would you suggest meeting this requirement?

Include: LOCK TABLE name IN SHARE MODE.

Ref: Many students will tend to suggest repeatable read as a solution to this requirement. However, repeatable read does not obtain and hold row locks until the row is accessed. It is therefore possible for other applications to change data that has not yet been read by the report application. The only way to guarantee the stability of an entire table is to lock the table before processing the rows.

- ___ 6. You receive several phone calls from users of your database complaining about a high number of deadlocks and timeouts. You begin investigating by taking a snapshot with the monitor and discover a high number of lock escalations. You know that you have configured the locklist and maxlocks parameters to handle your normal application requirements. What would be your next area of investigation?
-

Look for programs not committing frequently.

Ref: An inappropriate number of lock escalations can be caused by a lock list that is not large enough to handle the application load or a maxlocks parameter that does not support the largest unit of work generally required by some applications.

Assuming that these areas have been eliminated as a possible source, the most likely cause is an application that exceeds the normal design case concerning lock requirements. Students may also respond with other valid arguments as well. Again, the absolute answer is not important. The assumptions and thought process to arrive at a conclusion are the key.

Unit 9 Checkpoint

- ___ 1. What is the database manager configuration parameter that defines the level of error logging done to the administration notification log?
-

NOTIFYLEVEL

- ___ 2. What tools can be used to monitor DB2 activity?
-

Snapshot Monitor, Event Monitor, Health Monitor

- ___ 3. What type of tool is used to determine the access strategy DB2 will use to execute an SQL statement?
-

Explain

Unit 10 Checkpoint

- ___ 1. You are the only DBA permitted to create new packages on the production system. You are called at 2:00 a.m. by a member of the operations staff, who informs you that an application is ending with an SQLCODE of -805. What is your response?
-

Analyze the function of the new application and BIND the new package if justified.

Ref: Since it is given that the student is the only DBA with the authority to add new packages to the system, it follows that some degree of security is implied. The -805 code indicates that the package does not exist. Therefore, a new application is being put into production or the package for an existing application has been dropped. In either case, examining the function of the application is warranted.

-
2. Assume that there is another DBA with the authority to BIND any existing application. Three nights after your previous call from the operations staff, you receive another call. This time, the SQLCODE is a -818. What would you advise the DBA on duty to do?
-

Precompile, Bind, Compile, and Link the application.

Ref: The -818 indicates a consistency token (timestamp) mismatch between the executable and the package. In this particular case, several responses are valid. If the DBA has the capability to *trace* the preparation steps that have been completed, it may be possible to determine the specific steps required to resolve the problem. For example, perhaps all that is required is to bind the package from a bind file. The answer provided assumes that the *trace* capability does not exist and also favors the easy solution.

3. Chris is a programmer who will automate the creation of backups for your DB2 system. In a meeting to determine the proper interface, she asks for your advice on how to proceed. What do you tell her?
-

Use an API.

4. Mike is a programmer who will write an application to support part of a point-of-sales application. His program will need to check on the credit history of customers attempting to use a credit card. The input to the program is the credit card number. The credit history is in a 40-million row DB2 table, which has an index by credit card number. What interface would you suggest?
-

Embedded Static SQL.

Ref: Also, it is assumed that this transaction will be requested on a repetitive basis and that dynamic bind would not be desirable.

5. Your consulting services are requested at a planning meeting of a major software vendor. The software product being discussed must be able to perform ad hoc queries against a number of different database products. In order to prevent having to provide specific code for each database product supported, you contribute a suggestion. What is it?
-

Use CLI.

Ref: Also, a student may also suggest ODBC. This is a valid answer as well, but is targeted by the next question.

- ___ 6. The software application described in the prior question is going to be implemented using Microsoft Windows. What architecture might you recommend?
-

ODBC

Ref: Also, ODBC is not limited only to the Windows environment, but historically has been a solution popular with Windows developers. Visigenic Software Inc. provides an ODBC Software Development Kit for non-Microsoft platforms. See the Call Level Interface Guide for details.

- ___ 7. As the database administrator, you have been constantly requested to supply samples of production data for a test machine. The test machine has enough security so that the content of the data can be a sampling of actual production data; however, only 5% of the data needs to be periodically sampled for the test environment. In the past, you have manually retrieved the sample and would like to eliminate this task from your busy schedule. You would like staff leaders from the programming group to be able to request a random 5% sampling of any table for which they have SELECT authority. The application supporting this function would sample the data and optionally transmit the data to a target machine. How would you suggest implementing this application?
-

Use a stored procedure and dynamic SQL.

Ref: Since sampling the data would require server-intensive activity, the stored procedure approach is desired.

- ___ 8. A programmer is writing an application against a DB2 database. Some of the columns desired and the predicate is not totally known until execution time. What would you advise the programmer to use?
-

Dynamic SQL.

- ___ 9. You are attempting to determine the best cluster sequence for a large customer table. During your initial analysis, you have determined that online access to the table is generally via a unique CUSTNUMB. You also have many *batch* applications that require data to be sorted on LASTNAME. There are indexes on both of these columns. Given these brief requirements, what would you choose as the index to specify when performing a REORG, if either?
-

The index on LASTNAME.

Ref: Students may debate the correct answer to this question based on assumptions they may make concerning the access to this table. The response given reflects most accesses to data via unique keys. Clustering the data by such a key does not improve performance because access tends to be via equal predicates. There doesn't tend to be an inherent relationship between rows with *adjacent* values for the unique key. (Of course, there are exceptions to this general rule.)

The LASTNAME column has been indicated as one necessary to support ordering requirements, which in turn indicates that more than one data row will tend to be accessed. If the data is in cluster sequence according to this column, DB2 may choose to use the index to avoid actual sorts and also limit the I/O required to retrieve the data.

-
- ___ 10. The table analyzed for the prior question is examined more closely. Many online applications examine the data according to a range of ENTRY_DATE values. There is an index on ENTRY_DATE. Would you change your choice of cluster?

Yes. Change the clustering according to the index on ENTRY_DATE.

Ref: Supporting this answer is the assumption that MANY online applications access the data by a range of values against the ENTRY_DATE column. By changing the clustering of the table, the online access for these applications could be improved by reducing the I/O requirements of access. Of course, the batch applications mentioned in the prior question that require sorting on LASTNAME may be negatively impacted. However, the DBA should generally favor online performance over batch applications. Certainly, there are exceptions to this general rule.

- ___ 11. An application programmer asks for your advice concerning a program that produces a simple business report from the table previously described. The program accesses the server from a client workstation and appears to run for longer than necessary. It simply FETCHES about 20% of the data and writes the rows to a report. EXPLAIN indicates that the index on ENTRY_DATE is being used to support a range predicate in the query. The programmer assures you that the same statement runs fine in CLP, but takes much longer inside the static program. What would be a PRECOMPILE / BIND parameter that you should examine as one, if not the first, possible cause of the problem?
-

Blocking. Determine if it is enabled.

Ref: Certainly, students may respond with other plausible causes of the problem. However, the symptoms described indicate a strong possibility that blocking (or lack of it) could be the problem.

- ___ 12. A programmer presents you with an application that has a very complex SQL statement that accesses many tables. Assume that the statement is logically correct and that your database design is proper. However, the programmer implies that the strategy externalized by EXPLAIN accesses the *wrong* table first. Assume that you have plenty of time and CPU for program preparation. What PRECOMPILE / BIND parameter might you experiment with to influence DB2?
-

QUERYOPT

Ref: A basic assumption for this answer is that the installation does not always use a QUERYOPT level that would cause DB2 to consider all possible access strategies.

- ___ 13. You have just finished analyzing the SQL in a package that has been identified as a performance problem. You determined that an index to support the application was required, so you created the index. Assume that the new index was also based on the columns of the table by which the data should be clustered. What are your next three steps required for the package to take advantage of the new index?
-
-
-

REORG RUNSTATS REBIND

Ref: The REORG puts the table in the correct sequence according to the key specified, which is assumed to be the new index. The RUNSTATS command would gather the statistics for the new index and physical structure of the data, which is required for the REBIND to function properly.

Unit 11 Checkpoint

- ___ 1. Can the right to grant table, view, or schema privileges to others be extended to other users using the WITH GRANT OPTION on the GRANT statement?
-

Yes.

Appendix A Checkpoint

- ___ 1. What are the data types for large objects?
-

BLOB, CLOB, and DBCLOB

Index

A

access control 11-8
 access strategy 10-52
 access to database objects 11-20, 11-34
 administration authorities 11-10
 Administration Tools 1-9, 1-10
 Advisors 2-27
 Alert Center 1-9
 ALTER 11-12
 APIs 10-14
 application process 10-39
 Applications 1-10
 archival logging 7-19
 aslheapsz 10-65
 asynchronous page cleaner 10-61
 authorities 11-10, 11-12
 authority 11-8
 authorization for backup 7-40
 avg_apps 10-44

B

backup considerations 7-40
 Backup Database GUI 7-46
 backup file naming conventions 7-51
 backup, offline versus online 7-48
 BIND 10-70, 10-77, 11-12
 bind 10-26, 10-35
 bind command syntax 10-35
 BIND options 11-31
 11-31
 BINDADD 11-12
 BLOBs 5-51
 blocking 10-65
 BUFFPAGE 10-61
 buffpage 10-44

C

candidates for table space backup strategy 7-59
 catalog directory 4-63
 catalog views, packages 10-37
 change directory 4-63
 check constraints - definition 5-42
 check constraints - the need 5-41
 CHNGPGS_THRESH 10-61
 circular logging 7-17
 CLOBs 5-51
 CLP 1-9
 Command Center 1-9, 1-10, 2-21
 Command Line Processor 1-9
 compile 10-26

Configuration Assistant 2-34, 2-36
 configuration files 3-3
 configuration parameters for logging 7-27
 Configuration Tool 1-9
 CONNECT 11-12
 consistency token 10-26, 10-28
 CONTROL 11-12
 Control Center 1-9, 1-10, 2-19
 coordinated universal time (CUT) 7-69
 cpuspeed 10-44
 CREATE BUFFERPOOL statement 4-23
 create table 5-37
 Create Table Space Wizard 4-37
 create view 5-23
 CREATETAB 11-12
 CS 1-12
 CURRENT QUERY OPTIMIZATION special register 10-46, 10-48
 cursor stability 8-20

D

DAS 2-42
 data fragmentation 10-82
 data placement considerations 4-57
 data processing problems 7-6
 database 3-3, 4-4
 database access 11-20, 11-34
 database configuration file 3-3
 Database Configuration files 3-3
 Database Configuration window 4-60
 database directory 4-63
 Database Engine 1-9
 Database Managed Space 4-30
 Database Manager APIs 10-14
 database manager configuration file 3-3
 Database Manager Configuration files 3-3
 database monitoring 9-24
 Database Services 1-9
 Database System Monitor authorization 9-27
 Database System Monitor basic information 9-29
 Database System Monitor commands 9-30
 Database System Monitor functional groups 9-29
 Database System Monitor interfaces 9-27
 Database System Monitor, when active 9-28
 database_consistent 7-24
 db cfg 3-3
 DB2 1-15
 DB2 Administration Server 2-42
 DB2 Communication Support 1-12
 DB2 configuration files 3-3
 DB2 Connect 1-12

DB2 Connect Enterprise Edition 1-14
DB2 Connect Personal Edition 1-14
DB2 Database Partitioning Feature 1-8
DB2 Enterprise - Extended Edition 1-7
DB2 Enterprise Edition 1-7
DB2 Enterprise Server Edition 1-8
DB2 Express Edition 1-7
DB2 Family 1-12
DB2 get authorizations 11-42
DB2 Personal Edition 1-7
DB2 Physical Environment 3-3
DB2 prep syntax 10-32
DB2 Product Components 1-9
DB2 utilities 6-13, 6-19, 6-21
DB2 Workgroup Edition 1-7
DB2 Workgroup Server Edition 1-7
db2diag.log 9-6, 9-16
DB2GROUPS 11-23
db2rbind 10-54
db2trc 9-50
db2untag 4-35
DBADM 11-8, 11-10, 11-12
DBCLOBS 5-51
dbm cfg 3-3
DBM Configuration 4-62
deadlock causes 8-45
deadlock definition 8-45
deadlock detection interval 8-45
deadlock detector 8-45
default database configuration 4-24
defining table spaces 4-36
DELETE 11-12
Development Center 1-10
DFT_EXTENT_SZ 4-32
Diagnostics
 data required 9-6
 db2diag.log 9-6
directories 4-63
Directory Tool 1-9
DMS 4-30
DMS table space - minimum requirements 4-35
DRDA Application Server 1-16
dynamic versus static SQL 10-12

E
enforcing business rules 5-45
enforcing data rules 5-41
escalation of locks 8-36
Event Analyzer 1-9
event monitoring 9-25
EXECUTE 11-12
EXECUTE option 11-31
EXPLAIN 9-54, 9-55
explicit authorization 11-20
EXPORT utility 6-7, 6-13

extent 4-32
EXTENTSIZEx 4-32

F
file reference variables A-5
First-failure data capture 9-11
fragmentation 10-82

G
get authorizations 11-42
GET MONITOR SWITCHES 9-30
GET SNAPSHOT 9-32
grant specification 11-31
grant/revoke scenarios 11-38
grants 11-24
group IDs 11-23
groupids 11-24
groups 11-23

H
Health Center 1-10, 2-30
health monitoring 9-25
history file for recovery 7-35

I
implicit authorization 11-34
implicit privileges scenarios 11-37
implicit rebind 10-52
IMPORT utility 6-7, 6-19, 6-21
independent trace facility 9-50
INDEX 11-12
index clustering 10-72
index statistics 10-74, 10-80
individual ids 11-24
inoperative 10-52
INSERT 11-12
instance 3-3, 4-4
instance owner privileges 11-8
isolation level 8-20

J
Journal 1-9, 1-10, 2-32

L
Large object types 5-51
large objects - data manipulation concern A-4
large objects - tokens A-5
large table space 4-30
Launchpads 2-27
License Center 1-10
link 10-26

list authorizations 11-42
 Load wizard 6-30
 LOBs 5-51
 lock conversion 8-34
 lock escalation 8-36
 lock isolation level 8-20
 lock mode compatibility 8-15
 lock modes for rows 8-13
 lock modes for tables 8-10
 LOCK TABLE statement 8-32
 lock wait 8-40
Locking
 explicit 8-30
 locking application scenarios 8-10
 locking at table level only 8-32
 locking influenced by temporary tables 8-20
 locking objects 8-7
 locking strategies 8-8
 locklist configuration parameter 8-36
 locktimeout configuration parameter 8-40
 log buffer 7-14
 log file location 7-24
 log file naming convention 7-71
 log retention logging 7-19
 logbufsz 7-27
 logfilsiz 7-27
Logging
 dual/mirrored 7-22
 logging and ACTIVATE DATABASE 7-19
 logging configuration parameters 7-27
 logging/backup requirements summary 7-76
 loghead 7-23
 logpath 7-24
 logprimary 7-27
 logretain 7-27
 logsecond 7-27

M

max_coordagents 10-59
 maxagents 10-59
 maxappls 10-59
 maxcagents 10-59
 maxlocks configuration parameter 8-36
 mincommit 7-27
 modes of locks for rows 8-13
 modes of locks for tables 8-10
 monitoring activation 9-28

N

newlogpath 7-24, 7-27
 nextactive 7-23
 NLEAF 10-72
 NLEVELS 10-72
 NOTIFYLEVEL 9-7

NUM_IOCLEANERS 10-61
 num_poolagents 10-59
 numdb 10-59

O

objects of locks 8-7
 optimization class 10-46, 10-48
 optimizer functions 10-44
 optimizer, factors influencing 10-44
 OWNER option on BIND 10-31

P

package 10-52
 packages 10-28
 packages catalog views 10-37
 PAGEFREESPACE 6-55
 pckcachesz 10-26
 PCTFREE 6-55
 PDE 1-12
 performance 10-82
 performance benefits of dynamic SQL 10-12
 performance benefits of static SQL 10-12
 Physical Environment 3-3
 point-in-time roll forward 7-69
 precompile 10-26
 precompile syntax 10-32
 precompiler options 10-32
 prep 10-26
 prep syntax 10-32
 privilege 11-8
 privileges 11-12, 11-24
Problem
 description 9-5
 solving 9-4
 product components 1-12
 program preparation steps 10-26
 programming education 10-92

Q

QUALIFIER option on BIND 10-31
 query authorizations 11-42
 query privileges 11-42

R

read stability 8-20
 REBIND 10-54
 rebinding 10-52
 REC_HIS_RETENTN 7-35
 recovery history file 7-35
 recovery methods 7-12
 Recovery Tool 1-9
 REFERENCES 11-12
 referential integrity 5-37

regular table space 4-30
relational extenders A-7
REORG 10-70, 10-77
REORG TABLE 10-82
reorganize check 10-70, 10-77, 10-80
REORGCHK 10-70, 10-77, 10-80
repeatable read 8-20
Replication Center 1-10
RESET MONITOR 9-30
RESTORE command 7-55
Restore Database GUI 7-57
reuse of log files 7-71
revoke/grant scenarios 11-38
Roll Forward Database GUI 7-67
Roll Forward pending 7-61
rollforward 7-63
ROLLFORWARD options 7-69
ROLLFORWARD PENDING - clearing 7-69
row lock compatibility 8-15
row lock duration 8-20
RUNSTATS 10-70, 10-74, 10-77, 10-82
Run-Time Client 1-12

S

Script Center 1-9
SECADM 11-46
security listing 11-42
SELECT 11-12
sheapthres configuration parameter 10-56
SMS 4-30
SNA Support Feature 1-12
Snapshot monitoring 9-25
snapshots 9-32
softmax 7-27
sort passes 10-58
sortheap configuration parameter 10-56, 10-58
sorts - piped vs. non-piped 10-56
SQL Assist 2-23
statistics 10-70, 10-74, 10-77, 10-80
Stored Procedures 10-67
strict table locking 8-32
SYSADM 11-8, 11-10, 11-12
syscat.packages.valid 10-52
SYSCTRL 11-8, 11-10
SYSMAINT 11-8, 11-10
system authorities 11-10
system catalog tables 4-25
system catalog views 4-25
System Managed Space 4-30

T

table 4-4
table lock compatibility 8-15
table space 4-4

table space backup/restore considerations 7-59
table statistics 10-74, 10-80
Task Center 1-10
temporary table impact on locking 8-20
The 9-16
timeouts 8-40
timestamp 10-26
timestamps 10-28
Tool Settings 2-38
Tools Settings 1-9, 1-10
trace 9-50
Transaction logging 7-14
triggers - defining 5-48
triggers - the need 5-45

U

UDE 1-12
uncatalog directory 4-63
uncommitted read 8-20
unqualified references 10-30
updatable catalog views 10-85
UPDATE 11-12
UPDATE MONITOR SWITCHES 9-30
updating statistics manually 10-85
user-defined functions - defining A-16
user-defined functions - deterministic A-20
user-defined functions - external action A-20
user-defined functions - fencing A-20
user-defined functions - sourced on other functions A-18
user-defined types - defining A-12
user-defined types - the need A-11
userexit 7-27
utilities 6-13, 6-19, 6-21
utility 6-13

V

VALID 10-52
Visual Explain 1-9, 9-57

W

Wizards 2-27
writing logs 7-14

IBM[®]