IBM® DB2® Universal Database

Administration Guide: Implementation

Version 7

SC09-2944-01

IBM® DB2® Universal Database

Administration Guide: Implementation

Version 7

SC09-2944-01

Before using this information and the product it supports, be sure to read the general information under "Appendix M. Notices" on page 461.

Order publications through your IBM representative or the IBM branch office serving your locality or by calling 1-800-879-2755 in the United States or 1-800-IBM-4YOU in Canada.

Contents

About This Book ix
Who Should Use This Book x
How This Book is Structured x
A Brief Overview of the Other Volumes of the Administration Guide. xii
Administration Guide: Planning xii
Administration Guide: Performance . . . xiii

Part 1. Administering Using the Control Center 1

Chapter 1. Administering DB2 Using GUI Tools 3
Administration Tools 4
Common Tool Features. 6
Show SQL and Show Command 7
Show Related 7
Generate DDL. 8
Filter 9
Help 10
The Control Center. 11
Main Elements of the Control Center. . . 11
Using a Customized Control Center in DB2 for OS/390 12
Systems That Can Be Administered . . . 13
Objects that can be Administered 13
Displaying Systems in the Control Center 14
Managing DB2 for OS/390 Objects . . . 15
Adding DB2 for OS/390 Subsystems . . . 15
Managing Gateway Connections 16
Functions You Can Perform from the Control Center 16
Creating New Objects 17
Working with Existing Objects 17
Locating Objects (DB2 for OS/390 only). . 18
The Satellite Administration Center 18
The Command Center. 19
The Script Center 20
Using an Existing Script with the Script Center 21
Scheduling a Saved Command Script to Run 21
The Journal 21
Working with Jobs 22
The License Center. 22
The Alert Center 23
Client Configuration Assistant 23
Performance Monitor 24
Event Monitor 25
Using the Monitor Tools 26
Monitoring Performance at a Point in Time 28
Predefined Monitors 29
Action Required When an Object Appears in the Alert Center 31
Analyzing an Event for a Period of Time 31
Event Analyzer 32
Analyzing SQL Statements 34
Improving Performance of a Query . . . 34
Analyzing a Simple Dynamic SQL Statement 35
Managing Remote Databases 36
Managing Users. 38
Granting and Revoking Authorities and Privileges 38
Moving Data. 39
Managing Storage 41
Estimating Table and Index Size 41
Checking Available Space in a Table Space 42
Adding More Space to a Table Space. . . 43
Troubleshooting. 43
Replicating Data 44
Using Lightweight Directory Access Protocol 45
Using a Java Control Center 46
Running the Control Center as a Java Applet 46
Using Your Java Tools for Administration . . 46

Part 2. Implementing Your Design 49

Chapter 2. Before Creating a Database . . 51
Prerequisites Before Creating a Database . . 52
Starting DB2 52
Starting DB2 UDB on Windows NT . . . 53
Using Multiple Instances of the Database Manager 53
Organizing and Grouping Objects by Schema 54
Enabling Parallelism 55

Enabling Data Partitioning 57
Stopping DB2 59
Details on Creating a Database 60
Designing Logical and Physical Database Characteristics 61
Creating an Instance 61
License Management 70
Establishing the Environment Variables and the Profile Registry 70
Creating a DB2 Administration Server (DAS) 78
Creating a Node Configuration File . . . 95
Creating the Database Configuration File 97
Replicating Configuration Information Using Response Files 98
Enabling FCM Communications 98

Chapter 3. Creating a Database 101
Definition of Initial Nodegroups 103
Definition of Initial Table Spaces 103
Definition of System Catalog Tables 104
Definition of Database Directories 105
Local Database Directory 105
System Database Directory 106
Node Directory 106
DCE Directory Services 107
Lightweight Directory Access Protocol (LDAP) Directory Services 107
Creating Nodegroups 108
Definition of Database Recovery Log . . . 109
Binding Utilities to the Database 109
Cataloging a Database 109
Creating a Table Space 111
Creating a System Temporary Table Space 113
Creating a User Temporary Table Space 114
Creating Table Spaces in Nodegroups . . 114
Raw I/O 114
Creating a Schema 116
Setting a Schema 118
Creating and Populating a Table 118
Large Object (LOB) Column Considerations 121
Defining Constraints 123
Defining a Generated Column on a New Table 127
Creating a User-Defined Temporary Table 129
Defining an Identity Column on a New Table 130
Creating a Sequence 131
Comparing IDENTITY Columns and Sequences 133
Creating a Typed Table 133
Populating a Typed Table 133
Hierarchy Table 134
Creating a Table in Multiple Table Spaces 134
Creating a Table in a Partitioned Database 135
Creating a Trigger 136
Trigger Dependencies 138
Creating a User-Defined Function (UDF) or Method 138
Creating a Function Mapping 140
Creating a Function Template 140
Creating a User-Defined Type (UDT) . . . 142
Creating a User-Defined Distinct Type 142
Creating a User-Defined Structured Type 143
Creating a Type Mapping 143
Creating a View 144
Creating a Typed View 147
Creating a Summary Table 147
Creating an Alias 149
Creating a Wrapper 151
Creating a Server 152
Using Server Options to Help Define Data Sources and Facilitate Authentication Processing 153
Creating a Nickname 159
Referencing Nickname and Data Source Objects 160
Working with Nickname and Data Source Objects 160
Identifying Existing Nicknames and Data Sources 160
Creating an Index, Index Extension, or an Index Specification 161
Using an Index 165
Using the CREATE INDEX Statement . . 165
Creating a User-Defined Extended Index Type 168
Details on Index Maintenance 169
Details on Index Searching 169
Details on Index Exploitation 170
A Scenario for Defining an Index Extension 171

Chapter 4. Altering a Database 175
Before Altering a Database 175
Changing Logical and Physical Design Characteristics 175
Changing the License Information . . . 175

Changing Instances 175
Changing Environment Variables and the Profile Registry Variables 179
Changing the Node Configuration File 179
Changing the Database Configuration . . 179
Altering a Database 181
Dropping a Database 181
Altering a Nodegroup 182
Altering a Table Space 182
Dropping a Schema 188
Modifying a Table in Both Structure and Content 188
Altering a User-Defined Structured Type 204
Deleting and Updating Rows of a Typed Table 204
Renaming an Existing Table 204
Dropping a Table 205
Dropping a User-Defined Temporary Table 207
Dropping a Trigger 207
Dropping a User-Defined Function (UDF), Type Mapping, or Method 207
Dropping a User-Defined Type (UDT) or Type Mapping 208
Altering or Dropping a View 209
Dropping a Summary Table 210
Dropping a Wrapper 211
Altering or Dropping a Server 212
Altering or Dropping a Nickname . . . 213
Dropping an Index, Index Extension, or an Index Specification 214
Statement Dependencies When Changing Objects 215

Part 3. Database Security 219

Chapter 5. Controlling Database Access 221
Selecting User IDs and Groups for Your Installation 221
Windows NT Platform Considerations 223
UNIX Platform Considerations 224
General Rules 224
Selecting an Authentication Method for Your Server. 225
Authentication Considerations for Remote Clients 230
Partitioned Database Considerations . . . 230
Using DCE Security Services to Authenticate Users 231
How to Set up a DB2 User for DCE. . . 231
How to Setup a DB2 Server to Use DCE 233
How to Set up a DB2 Client Instance to Use DCE. 235
DB2 Restrictions Using DCE Security . . 235
Federated Database Authentication Processing 237
Authentication Settings 237
Passing User IDs and Passwords to Data Sources 238
Federated Database Authentication Example 240
Privileges, Authorities, and Authorization 242
System Administration Authority (SYSADM) 244
System Control Authority (SYSCTRL) . . 245
System Maintenance Authority (SYSMAINT) 246
Database Administration Authority (DBADM) 247
LOAD Authority 247
Database Privileges 248
Schema Privileges. 249
Table Space Privileges 250
Table and View Privileges 251
Nickname Privileges 253
Server Privileges 254
Package Privileges 254
Index Privileges 255
Sequence Privileges 255
Controlling Access to Database Objects . . 255
Granting Privileges 256
Revoking Privileges 257
Managing Implicit Authorizations by Creating and Dropping Objects 259
Establishing Ownership of a Plan or a Package 259
Allowing Indirect Privileges Through a Package 259
Allowing Indirect Privileges Through a Package Containing Nicknames 260
Controlling Access to Data with Views 261
Monitoring Access to Data Using the Audit Facility 264
Data Encryption 264
Tasks and Required Authorizations 266
Using the System Catalog 267
Retrieving Authorization Names with Granted Privileges 268
Retrieving All Names with DBADM Authority 268

Retrieving Names Authorized to Access a Table 268
Retrieving All Privileges Granted to Users 269
Securing the System Catalog Views . . . 270

Chapter 6. Auditing DB2 Activities . . . 273
Audit Facility Behavior 275
Audit Facility Usage Scenarios 277
Audit Facility Messages. 281
Audit Facility Record Layouts 282
Audit Facility Tips and Techniques 297
Controlling DB2 Audit Facility Activities . . 299

Part 4. Moving Data 303

Chapter 7. Utilities for Moving Data . . . 305

Part 5. Recovery 307

Chapter 8. Recovering a Database . . . 309

Part 6. Appendixes 311

Appendix A. Naming Rules 313
General Naming Rules 313
Object Naming Rules 313
Additional Information about Schema Names 314
Additional Information about Passwords 316
Using Delimited Identifiers in Object Names 317
How Case-Sensitive Values Are Preserved in a Federated System 317

Appendix B. Using Distributed Computing Environment (DCE) Directory Services . . 319
Creating Directory Objects 319
Database Objects 320
Database Locator Objects 321
Routing Information Objects 322
Attributes of Each Object Class 324
Details About Each Attribute 325
Directory Services Security. 329
Configuration Parameters and Registry Variables. 331
CATALOG and ATTACH Commands, and the CONNECT Statement 332
CATALOG GLOBAL DATABASE Command 332
CONNECT Statement 333
ATTACH Command 333
How a Client Connects to a Database . . . 333
Connecting to Databases in the Same Cell 335
Connecting to a Database in a Different Cell 336
How Directories Are Searched 337
ATTACH Command 337
CONNECT Statement 338
Temporarily Overriding DCE Directory Information 339
Directory Services Tasks 340
DCE Administrator Tasks 340
Database Administrator Tasks. 341
Database User Tasks 342
Directory Services Restrictions 343

Appendix C. User Exit for Database Recovery 345

Appendix D. Issuing Commands to Multiple Database Partitions 347
Commands 347
Command Descriptions 348
Specifying the Command to Run. . . . 349
Running Commands in Parallel on UNIX-Based Platforms 350
Monitoring rah Processes on UNIX-Based Platforms 351
Additional rah (Run All Hosts) Information (Solaris and AIX Only) . . . 352
Prefix Sequences 352
Specifying the List of Machines 355
Eliminating Duplicate Entries from the List of Machines 355
Controlling the rah Command 356
$RAHDOTFILES on UNIX-Based Platforms 357
Setting the Default Environment Profile on Windows NT 358
Determining Problems with rah on UNIX-Based Platforms 359

Appendix E. How DB2 for Windows NT Works with Windows NT Security . . . 361
A Sample Scenario with Server Authentication: 362
A Sample Scenario with Client Authentication and a Windows NT Client Machine:. 362

A Sample Scenario with Client Authentication and a Windows 95 Client Machine: 363
Using a Backup Domain Controller with DB2 363
User Authentication with DB2 for Windows NT. 364
User Name and Group Name Restrictions 364
DB2 for Windows NT Security Service 365
Installing DB2 on a Backup Domain Controller 365
Authentication With Groups and Domain Security 366

Appendix F. Using the Windows NT Performance Monitor 369
Registering DB2 with the Windows NT Performance Monitor 369
Enabling Remote Access to DB2 Performance Information 370
Displaying DB2 and DB2 Connect Performance Values 370
Accessing Remote DB2 Performance Information 372
Resetting DB2 Performance Values 372

Appendix G. Working with Windows NT or Windows 2000 Database Partition Servers 375
Listing Database Partition Servers in an Instance 375
Adding a Database Partition Server to an Instance 375
Changing the Database Partition 377
Dropping a Database Partition From an Instance 378

Appendix H. Configuring Multiple Logical Nodes 381

Appendix I. High Speed Inter-Node Communications 383
High Speed Interconnection Using TCP/IP 384
Prerequisites for Using an IBM Netfinity SP Switch 384
High Speed Interconnection Using VI . . . 385
Virtual Interface (VI) Hardware Setup . . 386
Enabling DB2 to Run Using VI 393

Appendix J. Lightweight Directory Access Protocol (LDAP) Directory Services . . . 395
Supporting LDAP Client and Server Configurations 395
Support for Windows 2000 Active Directory. 396
Configuring DB2 to Use Active Directory 397
Configuring DB2 in the IBM LDAP Environment 397
Creating an LDAP User 398
Configuring the LDAP User for DB2 Applications 399
Registration of DB2 Servers After Installation 399
Update the Protocol Information for the DB2 Server. 401
Catalog a Node Alias for ATTACH 401
Deregistering the DB2 Server 402
Registration of Databases 402
Attaching to a Remote Server 402
Deregistering the Database. 403
Refreshing LDAP Entries in Local Database and Node Directories 403
Searching 405
Registering Host Databases 405
Setting DB2 Registry Variables at the User Level 407
Enabling LDAP Support After Installation is Complete 407
Disabling LDAP Support 408
LDAP Support and DB2 Connect 408
Security Considerations 408
Security Considerations for Windows 2000 Active Directory 409
Extending the Directory Schema with DB2 Object Classes and Attributes 410
Extending the Directory Schema for IBM eNetwork Directory Version 2.1 411
Extending the Directory Schema for Windows 2000 Active Directory 411
DB2 Objects in the Windows 2000 Active Directory. 413
Object Classes and Attributes Used by DB2 413

Appendix K. Extending the Control Center 427
Performance Considerations 427
Packaging Considerations 427
Interface Descriptions 427
CCExtension 428
CCObject 429

CCMenuAction 432
CCToolBarAction 432
Usage Scenario. 433
MyExtension.java 434
MySample.java. 434
MyDatabaseActions.java 435
MyInstance.java 435
MyDB2.java. 436
MyDatabases.java 437
MySYSPLAN.java. 437
MyTable.java 438
MyDBUser.java 439
MyToolbarAction.java 439
MyAlterAction.java 440
MyAction.java 440
MyDropAction.java 440
MyCascadeAction.java 441
MyCreateAction.java 441

Appendix L. Using the DB2 Library . . . 443
DB2 PDF Files and Printed Books 443
DB2 Information 443
Printing the PDF Books 452
Ordering the Printed Books 453
DB2 Online Documentation 454
Accessing Online Help 454
Viewing Information Online 456
Using DB2 Wizards 458
Setting Up a Document Server 459
Searching Information Online 460

Appendix M. Notices 461
Trademarks 464

Index 467

Contacting IBM 477
Product Information 477

About This Book

The Administration Guide in its three volumes provides information necessary to use and administer the year 2000 ready, DB2* relational database management system (RDBMS) products, and includes:

- Information about database design (found in *Administration Guide: Planning*)
- Information about implementing and managing databases (found in *Administration Guide: Implementation*)
- Information about configuring and tuning your database environment to improve performance (found in *Administration Guide: Performance*).

Many of the tasks described in this book can be performed using different interfaces:

- The **Command Line Processor**, which allows you to access and manipulate databases from a graphical interface. From this interface, you can also execute SQL statements and DB2 utility functions. Most examples in this book illustrate the use of this interface. For more information about using the command line processor, see the *Command Reference*.
- The **application programming interface**, which allows you to execute DB2 utility functions within an application program. For more information about using the application programming interface, see the *Administrative API Reference*.
- The **Control Center**, which allows you to graphically perform administrative tasks such as configuring the system, managing directories, backing up and recovering the system, scheduling jobs, and managing media. The Control Center also contains `Replication Administration` to graphically set up the replication of data between systems. Further, the Control Center allows you to execute DB2 utility functions through a graphical user interface. There are different methods to invoke the Control Center depending on your platform. For example, use the `db2cc` command on a command line, (on OS/2) select the Control Center icon from the DB2 folder, or use start panels on Windows platforms. For introductory help, select **Getting started** from the **Help** pull-down of the Control Center window. The **Visual Explain** and **Performance Monitor** tools are invoked from the Control Center.

There are other tools that you can use to perform administration tasks. They include:

- The `Script Center` to store small applications called scripts. These scripts may contain SQL statements, DB2 commands, as well as operating system commands.

- The `Alert Center` to monitor the messages that result from other DB2 operations.
- The `Tool Settings` to change the settings for the Control Center, Alert Center, and Replication.
- The `Journal` to schedule jobs that are to run unattended.
- The `Data Warehouse Center` to manage warehouse objects.

Who Should Use This Book

This book is intended primarily for database administrators, system administrators, security administrators and system operators who need to design, implement and maintain a database to be accessed by local or remote clients. It can also be used by programmers and other users who require an understanding of the administration and operation of the DB2 relational database management system.

How This Book is Structured

This book contains information about the following major topics:

Administering Using the Control Center

- Chapter 1. Administering DB2 Using GUI Tools, introduces the graphical user interface (GUI) tools used to administer the database.

Implementing Your Design

- Chapter 2. Before Creating a Database, describes the prerequisites needed before creating a database and the objects within a database.
- Chapter 3. Creating a Database, decribes the tasks associated with creating a database and the objects within a database.
- Chapter 4. Altering a Database, describes the prerequisites and the tasks associated with altering or dropping a database and the objects within a database.

Database Security

- Chapter 5. Controlling Database Access, describes how you can control access to your database's resources.
- Chapter 6. Auditing DB2 Activities, describes how you can detect and monitor unwanted or unanticipated access to data.

Moving Data

- Chapter 7. Utilities for Moving Data, describes the Load, AutoLoader, Import and Export utilities.

Recovery

- Chapter 8. Recovering a Database, describes factors to consider when choosing database and table space recovery methods. Recovery tasks include backing up and restoring a database or table space, and using roll-forward recovery.

Appendixes

- Appendix A. Naming Rules, presents the rules to follow when naming databases and objects.
- Appendix B. Using Distributed Computing Environment (DCE) Directory Services, provides information about how you can use DCE Directory Services.
- Appendix C. User Exit for Database Recovery, discusses how user exit programs can be used with database log files and describes some sample user exit programs.
- Appendix D. Issuing Commands to Multiple Database Partitions, discusses the use of the *db2_all* and *rah* shell scripts to send commands to all partitions in a partitioned database environment.
- Appendix E. How DB2 for Windows NT Works with Windows NT Security, describes how DB2 works with Windows NT security.
- Appendix F. Using the Windows NT Performance Monitor, describes the two performance monitors available to DB2 for Windows NT users: the DB2 Performance Monitor, and the Windows NT Performance Monitor.
- Appendix G. Working with Windows NT or Windows 2000 Database Partition Servers, describes the utilities used by Windows NT and Windows 2000 to work with partitioned database servers.
- Appendix H. Configuring Multiple Logical Nodes, describes how to configure multiple logical nodes in a partitioned database environment.
- Appendix I. High Speed Inter-Node Communications, describes how to enable Virtual Interface Architecture for use with DB2 Enterprise - Extended Edition in the Windows NT environment.
- Appendix J. Lightweight Directory Access Protocol (LDAP) Directory Services, provides information about how you can use LDAP Directory Services.
- Appendix K. Extending the Control Center, provides information about how you can extend the Control Center by adding new tool bar buttons including new actions, adding new object definitions, and adding new action definitions.
- Appendix L. Using the DB2 Library, provides information about the structure of the DB2 library, including wizards, online help, messages, and books.

A Brief Overview of the Other Volumes of the Administration Guide

Administration Guide: Planning

The *Administration Guide: Planning* is concerned with database design. It presents logical and physical design issues; distributed transaction issues; and high availability topics. The specific chapters and appendixes in that volume are briefly described here:

The World of DB2 Universal Database

- ″Administering DB2 Universal Database″ presents an introduction to, and an overview of, DB2 Universal Database.

Database Concepts

- ″Basic Relational Database Concepts″ presents an overview of database objects, including recovery objects, storage objects, and system objects.
- ″Federated Systems″ discusses federated systems, which are database management systems (DBMSs) that support applications and users submitting SQL statements referencing two or more DBMSs or databases in a single statement.
- ″Parallel Database Systems″ provides an introduction to the types of parallelism available with DB2.
- ″About Data Warehousing″ provides an overview of data warehousing and data warehousing tasks.
- ″About Spatial Extender″ introduces Spatial Extender by explaining its purpose and discussing the data that it processes.

Database Design

- ″Logical Database Design″ discusses the concepts and guidelines for logical database design.
- ″Physical Database Design″ discusses the guidelines for physical database design, including considerations related to data storage.

Distributed Transaction Processing

- ″Designing Distributed Databases″ discusses how you can access multiple databases in a single transaction.
- ″Designing for Transaction Managers″ discusses how you can use your databases in a distributed transaction processing environment, such as CICS.

High Availability Systems

- ″Introducing High Availability and Failover Support″ presents an overview of the high availability failover support that is provided by DB2.

Appendixes

- "Planning Database Migration" describes information about migrating databases to Version 7.
- "Incompatibilities Between Releases" describes the incompatibilities introduced from release to release up to, and including, Version 7.
- "National Language Support (NLS)" describes DB2 National Language Support, including information about countries, languages, and code pages.

Administration Guide: Performance

The *Administration Guide: Performance* is concerned with performance issues; that is, those topics and issues concerned with establishing, testing, and improving the performance of your application, and that of the DB2 Universal Database product itself. The specific chapters and appendixes in that volume are briefly described here:

Introduction to Performance

- "Elements of Performance" introduces concepts and considerations for managing and improving DB2 UDB performance.
- "Architecture and Processing Overview" introduces underlying DB2 Universal Database architecture and processes.

Tuning Application Performance

- "Application Considerations" describes some techniques for improving database performance when designing your applications.
- "Environmental Considerations" describes some techniques for improving database performance when setting up your database environment.
- "System Catalog Statistics" describes how statistics about your data can be collected and used to ensure optimal performance.
- "Understanding the SQL Compiler" describes what happens to an SQL statement when it is compiled using the SQL compiler.
- "SQL Explain Facility" describes the Explain facility, which allows you to examine the choices the SQL compiler has made to access your data.

Tuning and Configuring Your System

- "Operational Performance" describes an overview of how the database manager uses memory and other considerations that affect run-time performance.
- "Using the Governor" describes an introduction to the use of a governor to control some aspects of database management.
- "Scaling Your Configuration" describes some considerations and tasks associated with increasing the size of your database systems.

- ″Redistributing Data Across Database Partitions″ discusses the tasks required in a partitioned database environment to redistribute data across partitions.
- ″Benchmark Testing″ presents an overview of benchmark testing and how to perform benchmark testing.
- ″Configuring DB2″ discusses the database manager and database configuration files and the values for the configuration parameters.

Appendixes

- ″DB2 Registry and Environment Variables″ describes profile registry values and environment variables.
- ″Explain Tables and Definitions″ describes the tables used by the DB2 Explain facility and how to create those tables.
- ″SQL Explain Tools″ describes how to use the DB2 explain tools: db2expln and dynexpln.
- ″db2exfmt — Explain Table Format Tool″ describes how to use the DB2 explain tool to format the explain table data.

Part 1. Administering Using the Control Center

Chapter 1. Administering DB2 Using GUI Tools

DB2 Universal Database provides graphical user interface (GUI) tools to help you administer local and remote databases easily from one central location called the Control Center.

This chapter presents an overview of the DB2 Universal Database administration tools that are available to you and explains how you can use them to get your job done easily and efficiently. It also gives you a summary of the Java Control Center and how you can customize the Control Center to include your own Java-enabled tools.

This chapter provides information on:

- "Administration Tools" on page 4
- "Common Tool Features" on page 6
- "The Control Center" on page 11
- "The Satellite Administration Center" on page 18
- "The Command Center" on page 19
- "The Script Center" on page 20
- "The Journal" on page 21
- "The License Center" on page 22
- "The Alert Center" on page 23
- "Client Configuration Assistant" on page 23
- "Performance Monitor" on page 24
- "Managing Remote Databases" on page 36
- "Managing Users" on page 38
- "Moving Data" on page 39
- "Managing Storage" on page 41
- "Troubleshooting" on page 43
- "Replicating Data" on page 44
- "Using Lightweight Directory Access Protocol" on page 45
- "Using a Java Control Center" on page 46
- "Using Your Java Tools for Administration" on page 46

Administration Tools

The tools for administering DB2 are part of the Administration Client, a selectable component with most of the DB2 Universal Database products. The Administration Client is also available on a set of CD-ROMs that include the Administration Clients for all the operating systems on which DB2 is available. They allow you to install and use the Administration Client on any workstation: It does not matter whether your database servers are local or remote, or what operating system the database servers are running on. The tools enable you to perform the same functions from a Graphical User Interface as you could from the Command Line Processor. These functions include the entering of DB2 commands, SQL statements, or system commands. With the tools, however, you do not have to remember complex statements or commands and you get additional assistance.

Note: The Administration Client is an installation option.

The following tools are available from the Control Center toolbar:

- The Control Center. The Control Center is the main DB2 graphical tool for administering your database. From the Control Center, you get a clear overview of all the systems and database objects that are cataloged locally.
- The Satellite Administration Center. The Satellite Administration Center allows you to administer DB2 Satellite servers.
- The Command Center. The Command Center enables you to issue DB2 database commands, SQL statements, and operating system commands; recall previous commands; and scroll through access plans for SQL queries.
- The Script Center. The Script Center enables you to create, run and schedule operating system commands, DB2 command scripts, and SQL statement scripts.
- The Alert Center. The Alert Center notifies you when thresholds that you have set have been exceeded or when a node in a multinode environment is no longer responding.
- The Journal. The Journal allows you to view the status of jobs, reschedule jobs, and to view the recovery history log and messages log.
- The Information Center. The Information Center gives you quick access to the information in the DB2 product manuals and sample programs and provides access to other sources of DB2 information on the Web.
- The License Center. The License Center displays the status of your license as well as allowing you to configure your system for proper license monitoring.

For some functions that you can perform with the GUI tools, you are given the option of using a Wizard. Wizards are invoked from the pop-up menus in the Control Center. They provide a greater level of help by prompting you

step-by-step on how to fill in the information necessary for the task you are doing and even making calculations and recommendations based on information you supply. Wizards are very useful if you are a new database administrator or someone who only administers a database occasionally.

In DB2 Universal Database, the following Wizards exist:

- Backup Database. This asks you basic questions about the data in the database, the availability of the database, and recoverability requirements. It then suggests a backup plan, creates the job script, and schedules it. To invoke the Backup Database Wizard, select the icon representing the database you want to backup, click the right mouse button, and select **Backup —> Database using Wizard**.
- Create Database. This Wizard allows you to create a database, assign storage, and select basic performance options. To invoke the Create Database Wizard, select the Databases icon, click the right mouse button, and select **Create —> Database using Wizard**.
- Create Table. This Wizard helps you to design columns using predefined column templates, create a primary key for the table, and assign the table to one or more table spaces. To invoke the Wizard, select the Tables icon, click the right mouse button, and select **Create —> Table using Wizard**.
- Create Table Space. This Wizard lets you create a new table space and set basic storage and performance options. To invoke it, select the Table Space icon, click the right mouse button, and select **Create —> Table space using Wizard**.
- Create Index. Use this Wizard to determine which indexes to create or drop for a given set of SQL statements. The recommendations are based on the workload that you specify. To invoke the Create Index Wizard, select the Indexes folder, click the right mouse button, and select **Create —> Index using Wizard**.
- Performance Configuration. This Wizard helps you tune databases by requesting information about the database, its data, and the purpose of the system. It then recommends new configuration parameters for the database and instance and automatically applies them if you wish. To invoke this Wizard, select the icon for a database, click the right mouse button, and select **Configure using Wizard**.
- Restore Database. This Wizard walks you through the process of recovering a database. To invoke the Wizard, select the icon for a database, click the right mouse button, and select **Restore —> Database using Wizard**.
- Configure Multisite Update. This Wizard lets you configure databases to enable applications to update multiple sites simultaneously. Doing this is important when the data at all the sites must be consistent. To invoke this Wizard, select an instance, click the right mouse button, and select **Multisite Update —> Configure using Wizard**.

Note: Wizards do not exist for the DB2 for OS/390 subsystem.

Besides the graphical tools that you can invoke from the Control Center toolbar, there are some additional GUI tools that are not invoked directly from the Control Center toolbar.

- Performance Monitor. Performance Monitor is a tool to monitor DB2 objects such as instances, databases, tables, table spaces, and connections. You use this tool to detect performance problems and tune databases for optimum performance. The Performance Monitor is invoked as a selection on the pop-up menus in the Control Center.
- Event Monitor. Event Monitor is a tool that lets you analyze resource usage by recording the state of the database at the time specific events occur. An Event Monitor is created by typing `db2emcrt` from a DB2 command line.
- Event Analyzer. Event Analyzer is a tool that allows you to analyze the data collected by the Event Monitor. An Event Analyzer is invoked by typing `db2evmon` from a DB2 command line.
- Visual Explain. The visual explain function lets you view the access plan for SQL statements as a graph so that you can tune your SQL queries for better performance. Prior to Version 6, you used the Visual Explain tool to view the access plans. Now, Visual Explain is no longer a separate tool; the function is available on pop-up menus from various database objects in the Control Center, and also from the Command Center.

In addition to these tools, another useful tool for database administration that is not part of the Control Center is the Client Configuration Assistant. The Client Configuration Assistant is a tool that contains Wizards to help users set up clients to communicate with remote servers.

All these tools are described in greater detail later on. The following section gives an overview of features found in the tools.

Common Tool Features

The following features are available in several tools:

- Show SQL and Show Command
- Show Related
- Generate DDL
- Filter
- Help

Show SQL and Show Command

If a tool generates SQL statements, then the **Show SQL** push button will be available on the tool interface. Similarly, a tool that generates DB2 commands will have a **Show Command** push button available. Clicking one of these push buttons allows you to:

- See the SQL statements or DB2 commands that the tool generates based on the choices you made in the graphical interface. This information helps you to understand how the interface is working.
- Save the statements or commands as a script for future reuse. This capability saves you from having to retype the SQL statements or DB2 commands if you want to run the same statements or commands again. Once the SQL statements or DB2 commands have been saved in a script, you can schedule the script, or edit the script to make changes, or create similar scripts without having to retype the statements or commands.

To show the SQL statements or DB2 commands:

1. From the Control Center, go to a window or notebook for working with a tool that generates SQL statements or DB2 commands. The **Show SQL** push button or the **Show Command** push buttom is shown as available.
2. Click on the **Show SQL** push button or **Show Command** push button. The appropriate window opens.

Saving SQL statements and DB2 commands is particularly helpful if the SQL statements or DB2 commands are complex.

When you use the **Show Command** or **Show SQL** push buttons, you can either create new scripts which you can later edit, or you can close the dialog box to return to the original dialog to make changes. If you click the Create Script push button, the New Command Script window appears. There you can edit the SQL statements or the DB2 commands before saving the script.

Show Related

Show Related shows the immediate relationship between tables, indexes, views, aliases, triggers, table spaces, User Defined Functions, and User Defined Types. For example, if you select a table and you choose to show related views, you only see any views that are based directly on the table. You would not see any views that are based on the related views because those views were not created directly from the table.

Showing related objects helps you to:

- Understand the structure of the database.
- Determine what indexes already exist for a table.
- Determine what objects are stored in a table space.

- Know what other objects are related to an object and are therefore affected by any actions you may take. For example, if you want to drop a table with dependent views, **Show related** shows you which views will become inoperative.

To use the Show Related feature:

- From the Control Center, select an object from the Contents pane and click the right mouse button.
- Select **Show Related**
- Click on the tab to open the page for the related objects you want. Different related objects are listed depending on the tab you select. Only objects that are directly related to the object that you have selected are shown.

  You can click the right mouse button on a related object on the selected Page and select Show Related from the pop-up menu. The selected Page changes to show the objects related to your latest selection. You can also click on the down arrow next to the selected object to display a list of objects you previously selected to show relationships.
- Click **Close** to close the Show Related notebook and return to the Control Center.

Generate DDL

The **Generate DDL** function allows you to re-create and save in a script file the DDL and SQL statements and statistics of:

- Database objects
- Authorization statements
- Table spaces, nodegroups and buffer pools
- Database statistics

This allows you to:

- Save the DDL to create identically defined tables, databases, and indexes in another database, for example, for a database warehouse application
- Use the DDL to copy a database from the test environment to a production environment or from one system to another
- Edit the DDL to create similar objects

Clicking the **Generate DDL** push button brings up the Show Command window with statements generated by a utility known as the **db2look** utility. From the Show Command window, you can click the **Save Script** push button to save the statements. The statements are put into a script. If you click the **Generate** button, the Run Script window opens.

Note: Generating DDL statements is different when working with the Control Center for System 390. See the help information for specifics on those differences.

You can select whether you want to generate DDL statements for selected schemas or all schemas within the database. You can then edit the script if you want to make changes before you use the script in a production environment. To create identical databases using the generated DDL statements, you would simply use the script which you generated and run it in the new environment.

To generate DDL statements:

1. Highlight the object for which you want to generate DDL statements, and click the right mouse button.
2. Select **Generate DDL**. The Run script window appears.
3. Type a user ID and password, and click **OK**. A job is created with the contents of the **db2look** command. A DB2 message window appears with the job ID of the new job.
4. Click on **OK** to close the message window.
5. Use the Job History page of the Journal notebook to view the results of the job and to view the contents of a saved script associated with the job.
6. Select the job and click the right mouse button. Select **Show Results** from the pop-up menu. The Job Results window opens. The output of the **db2look** command is shown in the Job Output pane.
7. Select **Create Script** to create a script of the results. The New Command Script window appears.
8. Save the new script if you want to use it again.

Filter

In the Control Center, you can filter information that is displayed in the Contents pane, or you can filter information that is retrieved from a table as an actual result set. You can limit the number of objects that are displayed or the number of objects that are returned by creating filters for one or more objects. Once you have set the filter, you need to clear or delete the filter if you want to display all the objects in the tree once again.

Filtering the Display

To reduce the number of objects that appear in the Contents pane for more manageable administration:

1. Select the Filter icon from the Contents pane toolbar, located at the bottom of Control Center, or select Filter from the View menu bar.
2. Select the criteria to be used to reduce the number of objects.
3. Select the Enable filter checkbox to activate the filter.

When you later select an object to view its contents, the filter you have associated with the object limits the view according to the criteria you set earlier.

Filtering Retrieved Data

To reduce the number of rows returned in a query and improve the response time, you can define the output, or the result set that shows in the Contents pane when selecting an object.

1. Select a folder object from the tree and click the right mouse button.
2. From the pop-up menu, select **Filter**. The Filter window opens.
3. Use the Filter function to define a set of criteria for retrieving rows belonging to that object.

Defining a Filter to Retrieve a Specific Set of Data

To define a filter to retrieve a specific set of data:

1. From the Control Center, expand either the Databases or Subsystems folders depending on your platform.
2. Select an object for which you want to define the filter. Click the right mouse button on that object.
3. Select **Filter** from the popup menu. This opens the Filter notebook.
4. On the Locate page, specify the name or other descriptive filter criteria of the selected object. The result of the filter is the results set associated with the selected object shown in the Contents pane of the Control Center.
5. On the Locate page, select a radio button to specify whether to meet all the conditions selected in the fields on the Locate page or to meet at least one condition.
6. On the Advanced page of the Filter notebook, you can use additional criteria by editing the text that is shown to further limit the number of returned rows.
7. Click **OK** to use the filter criteria you defined.

To automatically invoke this filter notebook based on numbers of rows, select Tools from the menu bar, and select Tools Settings from the popup menu. The **Select filtering when numbers of row exceeds** checkbox allows you to predefine a threshold of returned rows from any selection. When the threshold is reached, the Filter notebook appears so that you can limit the current retrieval based on the defined criteria. This is especially useful when a table has grown unexpectedly and was previously unfiltered. Depending on your platform, and your data, you could be attempting to return millions of rows, when you need only a subset of rows.

Help

Extensive help information is provided with the administration tools. A help button exists on all windows and notebooks as well as on the menu toolbar. You can get general help as well as help on how to fill out the fields and perform tasks. From the help menus, you can also access the index of terms or the reference information and the information provided in the product manuals.

The Control Center

Use the Control Center as your main point of administration to manage systems, DB2 instances, databases, database objects, such as tables, views, and user groups. You can also use the Control Center to access DB2 for OS/390 subsystems. All DB2 databases must be cataloged before they appear in the Control Center. The Figure 1 shows the primary features of the Control Center. Because of operating system differences, the Control Center on your system may appear different from the diagram.

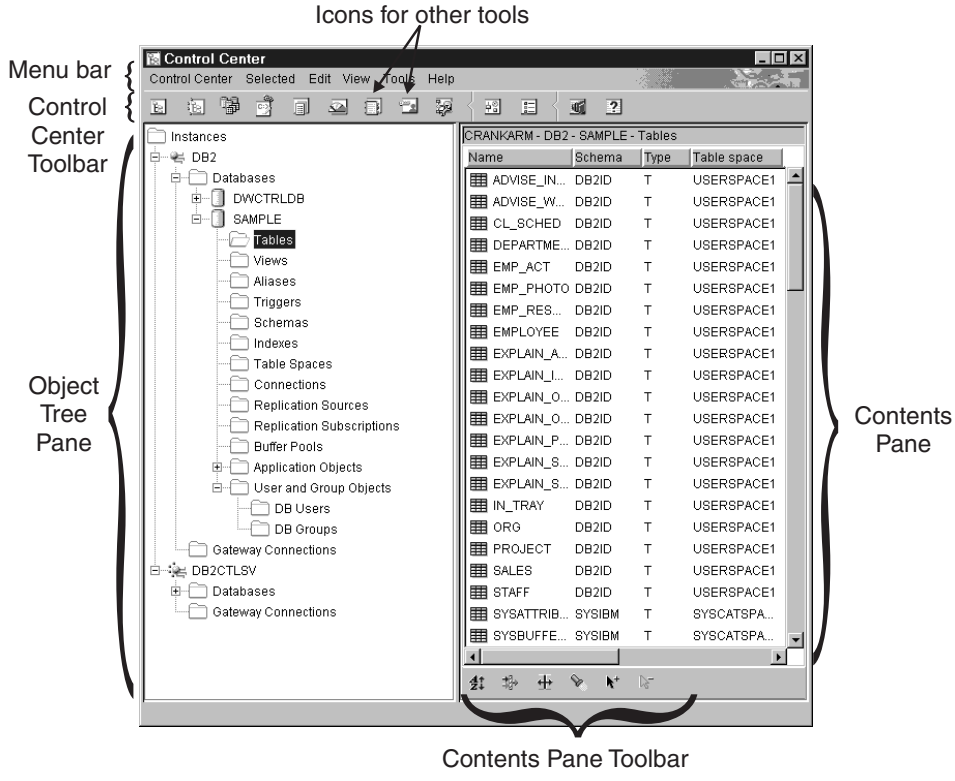


Figure 1. Control Center Features

Main Elements of the Control Center

The main elements of the Control Center are:

- Menu Bar. The menu bar is at the top of the screen. Selecting a menu from the menu bar allows you to perform many functions, such as shutting down the DB2 tools, accessing the graphical tools, and getting access to online help and product information. You should familiarize yourself with these functions by clicking on each item on the menu bar.

- Control Center Toolbar. Icons for the Control Center and the other tools are located on the Control Center toolbar. Hover help identifies each icon when the cursor is placed over the icon.

  You can change the settings for these tools by selecting the Tools Settings icon from the Control Center toolbar.
- Object Tree. The object tree is located on the left pane of the screen. It displays icons for all the database servers and objects that you can manage from the Control Center. You must first catalog a remote database server before it appears in the Object Tree pane. Some objects in the Object Tree pane contain other objects. A plus sign (+) to the left of the object indicates that the object is collapsed. You can expand it by clicking the plus sign. A minus sign (–) appears to the left of an object when it has been expanded. To collapse the object, click the minus sign.
- Contents Pane. The Contents pane is located on the right pane of the screen. This pane shows all the objects that are contained in the selected object in the Object Tree pane, for example, if you select the tables folder in the Object Tree pane, all your tables show up in the Contents pane. If you select the databases folder, the Contents pane changes to show all databases. You can filter the columns that appear in the Contents pane by clicking the Filter icon in the Contents pane toolbar and specifying the required information or you can filter objects by selecting **Tools** on the toolbar and then **Tools Settings**. You must ensure that the **Enable Filter** check box is selected in the Contents Pane filter dialog.
- Contents Pane Toolbar. This toolbar appears at the bottom of the Contents pane. It allows you to tailor the information in the Contents pane. This toolbar is a common control which appears at the bottom or to the side of most detailed views throughout the product.

When working in the Control Center you may see that some fields are highlighted by a bold red border. This border indicates that this is a mandatory field requiring input from you. Once you have selected a value or entered a value, the red border will disappear.

Using a Customized Control Center in DB2 for OS/390

Use the Customized Control Center on the DB2 for OS/390 platform as your own defined point of administration to manage subsystems, databases, or database objects, such as tables, views, and database users. You can use this Customized Control Center to access any DB2 for OS/390 objects that you define.

The main elements of the Customized Control Center are the same as those listed for the default Control Center. The Customized Control Center allows you to specify objects that you want to include in a personalized Control Center. This user-defined tree can be saved and invoked to administer DB2 objects. It does not replace the Control Center tree, which is the default for all

users, but is useful if you want to access a set of objects in the same way each time the Control Center is invoked. You can create as many customized trees as you need, and each one can contain a different set of objects, which can be ordered in any way that you choose.

Using a customized tree reduces the effort of navigating through a fixed hierarchy of DB2 objects, and provides a method of grouping related objects. For example, you can define a tree that contains only tables with payroll information.

Systems That Can Be Administered

From the Control Center, you can administer database objects for the DB2 Universal Database family of products for OS/2, Windows, and UNIX platforms. Refer to the *Quick Beginnings* books for your platform-specific installation and setup information.

You can also replicate data from DB2 for AS/400, DB2 for VSE and VM systems, and DB2 for OS/390 to the DB2 Universal Database family of products. Refer to the *Replication Guide and Reference* manual for information on replication between products.

Objects that can be Administered

If you want to administer objects from the Control Center, you must add them to the object tree. If you remove a database, or uncatalog it outside of the Control Center, and you want to use the Control Center to perform tasks on it, you must add it to the object tree.

The DB2 Universal Database objects that you can administer from the Control Center are:

- Systems
- Instances
- Tables
- Views
- Indexes
- Triggers
- User-defined types
- User-defined functions
- Packages
- Aliases
- Replication Objects
- Users and Groups

The DB2 for OS/390 Version 5 objects that you can administer from the Control Center are:

- Buffer Pools
- Views
- Catalog Tables
- Storage Groups
- Aliases
- Synonyms
- DB2 Users
- Locations
- Application Objects (Collections, Packages, Plans, Procedures)
- Databases
- Tables
- Table Spaces
- Indexes
- Replication Source
- Replication Subscriptions

For DB2 for OS/390 Version 6, the objects you can administer from the Control Center are all of the objects mentioned for Version 5, plus:

- Schemas
- Triggers
- User Defined Functions
- Distinct Types

To see what actions can be performed on each of these objects, select the object in the Object pane and click the right mouse button. A pop-up window appears listing the functions.

Displaying Systems in the Control Center

To display all of the systems that are cataloged on your system and which have DB2 installed:

1. Expand the Object Tree by clicking the plus sign (+) beside **Systems**. Icons representing the local machine and any remote machines are displayed. Your local system is represented by the Local icon. It only appears if the local machine is a DB2 server. If you click the right mouse button on the Local icon, one of the options in the pop-up menu is called **Attach to administration server**. The Administration Server lets you take advantage of functions such as performance monitoring and scheduling. It is used by the DB2 administration tools to satisfy DB2 service requests and it is automatically created and started for you. The default name for the DB2

Administration Server varies by platform. For example, on Windows and OS/2 platforms, "DB2DAS00" is used; on AIX "db2as" is used.

2. Expand the Local icon. The instance of DB2 on the local machine is displayed in a tree structure.

   On OS/2, Windows and supported DB2 UNIX-based systems, you can think of each copy of the database manager code as a separate *instance*, that is stored in a directory on your machine. On DB2 for OS/390, an instance is referred to as a subsystem. A default local instance is created when you install DB2. You can have several instances on a single system. You can use these instances to separate the development environment from the production environment, or to restrict sensitive information to a particular group of people. You can also tune an instance for a particular environment.

3. Expand the **Instances** icon. For each database that exists, an icon and the name are displayed.

Managing DB2 for OS/390 Objects

Using the Control Center, you can perform many of the functions of the existing DB2 for OS/390 Version 5 and DB2 UDB for OS/390 Version 6 products, such as creating, altering, and dropping objects, as well as run utilities that reorganize or load your data. However, before you can administer a DB2 for OS/390 subsystem from the Control Center, you must first add it to the object tree by configuring a connection to it.

Adding DB2 for OS/390 Subsystems

If you have the Client Configuration Assistant installed, you can use it to configure a connection to a DB2 for OS/390 system easily. If you do not have the Client Configuration Assistant installed, you will have to configure the connection to the DB2 for OS/390 system manually, using the Command Line Processor (CLP).

You use the Client Configuration Assistant to search the network for all the DB2 for OS/390 systems which are available on the LAN to your client. If you would like to add one of the DB2 for OS/390 systems, you can use the **Add Database** Wizard to add the system, import a connection by using a profile, or add the connection manually.

If you choose to search the network, you need to have a DB2 Connect product on your network with a connection defined for the system. If you choose to use an access profile, you need to select the DB2 Connect server connection that represents the system from the profile. If you choose to manually configure the connection, you need to know the system name, the communication protocol, and the communication protocol parameters such as the host name and the port number for TCP/IP, or the Symbolic Destination Name for SNA. Once you add the DB2 for OS/390 system, objects for DB2 Connect server connections will appear in the Control Center's local system.

When you add a DB2 for OS/390 Version 5 or later system, it appears in its own section of the Control Center object tree. To see the DB2 for OS/390 and other database objects that reside in a particular system, expand the object tree from the DB2 for OS/390 system icon that represents your DB2 for OS/390 system.

To see the list of actions that you can perform on a particular object, select the object as it appears in the object tree and click the right mouse button. A pop-up menu appears and shows the available actions you can perform on that object. For example, you can create, alter, or drop a view, as well as see its contents, modify the privileges on it, and show a list of other objects that are related to it. See the online help for the DB2 for OS/390 objects for more information on what functions you can perform.

Managing Gateway Connections

When a DB2 Connect server is cataloged, a Gateway Connections folder is displayed in the Control Center object tree under the instance object of the local system. The Gateway Connections folder contains a hierarchy of objects used to manage connections to host and AS/400 databases that are cataloged locally. The actions associated with these connection management objects can be used to list, force, and monitor host and AS/400 database connections.

The object tree under the Gateway Connections folder is used for managing connections to host and AS/400 databases but not for database administration tasks. However, if you need to add, change, or remove a host or AS/400 database on the local system, use the Client Configuration Assistant.

Functions You Can Perform from the Control Center

From the Control Center, you can:

- 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, reorganize data, and gather statistics.
- Schedule jobs. Jobs may be pending, running, or completed executions of scripts. You can schedule jobs to start at particular times.
- Perform preventive maintenance by backing up and restoring databases or table spaces.
- Monitor performance and perform troubleshooting.
- Replicate data.
- Configure and tune instances and databases.
- Manage database connections, such as DB2 Connect servers and subsystems. Manage applications.
- Analyze queries using Explain SQL to look at access plans.

- Change the font used for displaying Menus and Text throughout the Control Center. You can change to one of the available fonts, the size of font, and the color shown. The Control Center must be restarted for the changes to take effect.
- Launch other tools. For example, you can launch the Satellite Administration Center or the Command Center.

To see all the actions that you can perform on an object, simply select the object from the Object Tree pane or the Contents pane and click the right mouse button. A pop-up menu appears showing all the functions that you can perform on that type of object; for example, if you select the tables folder, you can create a new table with or without the help of a Wizard, monitor the performance of tables, filter which tables appear in the Contents pane and so on. The functions you can perform are different, depending on the object you select.

Click the right mouse button on the objects in the Contents pane to perform additional functions on a specific object. For example, if you select one of your tables in the Contents pane and click the right mouse button, a pop-up window displays functions you can use on that table.

Creating New Objects

To create new objects:

1. Expand the databases folder. Object types are displayed as folder icons.
2. Click the right mouse button on the folder icon for an object, for example, click on the **Tables** icon. The pop-up menu is displayed. For some objects, you get two options to perform a function. One option is to use the Wizard. Wizards do not exist for all functions that you can perform.
3. Select **Create**. Since there is a Wizard to create a table, you get two options, one of which is to create the table using the Wizard. If you choose the Wizard option, you are prompted for information and given suggestions on what choices you should make. The Wizard is especially useful for new users or people who create database objects infrequently.

Working with Existing Objects

When you click an object such as the table folder in the Object Tree pane, all tables already existing appear in the Contents pane. You can then select a table you want to work with and click the right mouse button to invoke any functions that you wish to perform on that specific table.

For more information about using the Control Center, go to its online help, available from its **Help** menu or by pressing F1 anywhere in the Control Center.

Locating Objects (DB2 for OS/390 only)

You can search for a database or subsystem object easily by using the Locate notebook. This allows you to:

- Find an object without having to navigate through the tree structure of the Control Center. The object could be in a database or subsystem, table space, or across databases and tables and supporting objects.
- Locate objects (table spaces, tables, and indexes) across multiple databases within a subsystem.

Use the Locate page of the Locate notebook to specify the search criteria. Use the Advanced page of the Locate notebook to further customize the search. Edit the text provided on the Advanced page and add or modify the search criteria.

To locate an object defined within a database or a DB2 for OS/390 subsystem:

1. From the Control Center, click the right mouse button on an object. Select **Locate** from the popup menu. The Locate notebook opens.
2. From the Object type field, select the type of database object to search. The list of target objects available varies, depending on the object from which you begin your search.
3. On the Locate page, fill in the search criteria. You must type in at least one search criterion and you can use wild cards to help in the search. Characters are folded to uppercase unless you use valid delimiters to enclose lower case characters or the extended character set.
4. On the Locate page, select a radio button to specify whether to meet all the conditions selected in the fields on the Locate page or meet at least one of the conditions.
5. Click **OK** to use the search criteria. The results of your search are displayed in the Locate Result window. The format of the output table depends on the type of object for which you searched.
6. To repeat the search with the same or different criteria, click **APPLY**.
7. You can select a row that appears in the Locate Result window and right click on that row to see a popup menu with additional actions that you can perform.

The Satellite Administration Center

The Satellite Administration Center is a set of tools that is available from the DB2 Control Center. They allow you to set up and administer collections of DB2 servers from a central point. Each DB2 server that belongs to a group is known as a satellite. Administering satellites from a central point means that DB2 can be hidden from anyone using a DB2 satellite, thereby avoiding the need for them to learn about database administration.

Use groups to organize DB2 servers that have shared characteristics, such as the applications that run on them or the database configuration that supports the application. The DB2 servers are similar in terms of their database configuration, usage, and purpose.

By grouping the DB2 servers together, you can administer groups of DB2 servers rather than having to administer each DB2 server individually. If you acquire additional DB2 servers to serve the same function as the DB2 servers of an existing group, you can add them to that group by using the Satellite Administration Center.

From the Satellite Administration Center, you can create groups, satellites, application versions, batches, and authentication credentials. You can also define success code sets and perform other functions associated with the administration of the satellite environment. Information about the satellite environment is stored in a central database known as the satellite control database. This database records, among other things, which satellites are in the environment, the group each satellite belongs to, and which version of the end-user application a satellite is running. This database is on a DB2 server that is known as the DB2 control server.

Before the functionality of the Satellite Administration Center can be enabled, you must first catalog a satellite control database (SATCTLDB) on the Control Center. When it is enabled, you can use the Satellite Control Center to set up and maintain satellites, groups, and the batches that the satellites execute when they synchronize for their application version.

To set up and maintain its database configuration, each satellite connects to the satellite control database to download the batches that correspond to its version of the end-user application. The satellite executes these batches locally, then reports the results back to the satellite control database. This process of downloading batches, executing them, then reporting the results of the batch execution is known as synchronization. A satellite synchronizes to maintain its consistency with the other satellites that belong to its group and are running the same version of the end-user application.

The Command Center

You can start the Command Center from the Control Center by clicking the Command Center icon on the toolbar.

The Command Center lets you:

- See the resulting output of one or many SQL statements and DB2 commands in a result window. You can scroll through the results and generate a report.

- Create, and save command scripts to the Script Center. You can edit the command script to create new scripts. From the Script Center, the command script can then be scheduled to run as a job at any time that you specify.
- Run SQL statements, DB2 commands and operating system commands. When you run DB2 commands from the Command Center, you do not have to precede the command by **DB2**. You can run operating system commands in any supported operating system script language, such as REXX, by preceding them with an exclamation mark (!). Using the Command Center to run the commands and statements allows you to issue many commands at once, without the need to type and run each command individually.
- Get quick access to the DB2 administration tools, such as the Control Center, from the main toolbar.
- See the access plan and statistics associated with an SQL statement before execution.

The Script Center

You can start the Script Center by selecting its icon from the Control Center Toolbar. The Script Center is a tool that allows you to create scripts by writing a set of commands and statements, which you can schedule to run whenever you require. You can import scripts that you created earlier or scripts that you saved in the Command Center. You can select scripts from the set of scripts saved and you can edit existing scripts to create new scripts, copy scripts, or remove scripts.

You can edit a script inside the Script Center or outside the Script Center using your own editor. If you run a script from within the Script Center, you get the added advantage of having the results logged in the Journal.

To run operating system commands from a script in the Script Center:

1. Select **Script —> New**. The New Command Script window opens.
2. For **Script Type**, select the **OS command** radio button.
3. Enter the script name, description, and working directory.
4. Enter the commands.
5. Click on **OK**.

From the Script Center you can view information such as the description and script type, about all command scripts that are known to the system and you can perform the following tasks:

- Create a command script that contains DB2 and operating system commands
- Run a saved command script immediately

- Schedule a script to run at a later date or at a regular interval; for example, you might want to create a script that collects statistics for several tables. You could then schedule the job to run overnight. You can schedule jobs by running them unattended at scheduled intervals by specifying the hours, days, weeks, months, multiple times a week, or multiple times a month you want to run the job. A job is created whenever you schedule a script or run a script immediately.
- Access the Journal from the toolbar to see the jobs that use a particular script and to see the status of all scheduled jobs
- Edit a saved command script

Using an Existing Script with the Script Center

To use the Script Center with pre-existing scripts which you did not create from the Script Center:

1. From the **Control Center** toolbar, click on the **Script Center** icon. The Script Center opens.
2. Select **Script —> Import**. The File Browser window opens.
3. Select an existing script file and click on **OK**. The New Command Script window opens. The script is displayed in the lower part of the window which is a script editor. Complete the **Instance**, **Script name**, **Script description** and **Working directory** fields, and select a **Script type**.
4. Click on **OK**. The script will be created in the Script Center.

Scheduling a Saved Command Script to Run

To schedule a script:

1. Click the **Script Center** icon on the Control Center toolbar. The Script Center opens.
2. Click with the right mouse button on the script you want to schedule to run and select **Schedule** from the pop-up menu. The Scheduler window opens.
3. Select the frequency for the job, and a completion action, such as a completion message or another command script to be launched.
4. Click **OK**. This starts a pending job that you can track in the Journal.

The Journal

You can start the Journal by selecting its icon from the Control Center toolbar. The Journal allows you to monitor jobs and review results. From the Journal, you can also display the recovery history and DB2 messages. The Journal allows you to:

- Monitor pending jobs, running jobs, and job histories
- Review results
- Display recovery history and alert messages

- Show the log of DB2 messages

Working with Jobs

Use the Journal to work with jobs. To open the Journal:

1. Click the **Journal** icon from the Script Center toolbar. The Journal opens.
2. To see the jobs that are scheduled to be run at a later time, click the **Pending jobs** push button. You see your job in the list of pending jobs. You also see all the information about the jobs. You can perform actions on a pending job, such as reschedule it, show the scripts associated with it, or run it immediately. When a saved script is modified, all jobs that are dependent on it inherit the new modified behavior.

From the Journal, you can also see the jobs that are currently running and the job histories.

The other pages in the Journal window are:

- The Recovery page. This page displays the recovery history (the details from backup, restore operations, and load operations) and lets you restore the recovery log.
- Alerts page. This page shows all alerts.
- The Messages page. This page shows all messages issued through the DB2 administration tools.

The online help for the Journal provides detailed steps for working with jobs and logs.

The License Center

You use the License Center to display license status and usage information for DB2 products installed on your system. You can also use the License Center to configure your system for proper license monitoring. The License Center allows you to:

- Add a new license.
- Upgrade from a trial license of the product to a permanent license.
- View the details of your license.

If you view the details of the license information, you see the:

- Product name
- Version information
- Expiry date
- Registered users
- Concurrent users
- Number of entitled users

- Concurrent number of users
- Enforcement policy
- Number of processors (for DB2 Universal Database Enterprise Edition and Enterprise – Extended Edition).

The Alert Center

The Alert Center is the tool that monitors your system to warn you about potential problems. You can set the Alert Center to automatically open to display any monitored objects that have exceeded their threshold and are therefore in a state of alarm or warning. You set up the thresholds using the Performance Monitor, which is invoked from the Control Center. The color of the icon indicates the severity of the warning. A red icon indicates an alarm. A yellow icon indicates a warning.

Client Configuration Assistant

The Client Configuration Assistant (CCA) is primarily a tool that contains wizards to help set up clients to local or remote DB2 servers. However, the tool can also be used to easily help configure DB2 Connect servers.

The Client Configuration Assistant lets you maintain a list of databases to which your applications can connect. It catalogs nodes and databases while shielding you from the inherent complexities of these tasks.

From the Client Configuration Assistant you can perform the following tasks:

- Add, modify, and delete database connection entries.
- Test the connection to a selected database.
- Configure database manager configuration parameters.
- Configure CLI/ODBC settings.
- Bind DB2 utilities and other applications to a selected database.
- Import and export configuration information. This allows you to use the existing configuration on a machine that was previously configured to configure new machines.
- Change the password for the user ID that you use to connect to a selected database.

The Client Configuration Assistant provides the following methods to assist in adding new database connection entries:

- Use a profile. A profile can be exported from a previously configured machine and used to configure new machines. Server profiles can be exported from the Control Center, and client or server profiles can be exported from the CCA.

- Search the network. The CCA can search the network for DB2 systems which have an administration server running. A Search and Known (or directed) Discovery mode is provided. Search Discovery mode is subject to network configuration restrictions. (Typically network routers will not allow a Search Discovery request to be transmitted.) Known Discovery requires only a few pieces of information to find the server system desired. Host or AS/400 systems that have been previously defined on a gateway, can also be found.
- Manually configure a connection to a database. All information must be provided, but a wizard is started to help make the task simpler.

Performance Monitor

The Performance Monitor provides information about the state of DB2 Universal Database and the data that it controls. It is a graphical utility that can be customized for your database environment. You can define thresholds or zones that trigger warnings or alarms when the values being collected by the Performance Monitor are not within acceptable ranges.

You can monitor DB2 objects such as instances, databases, tables, table spaces, and connections by selecting the object in the Object Tree pane or in the Contents pane and clicking the right mouse button. From there, you can choose to start monitoring activity.

When an object is being monitored, the color of the icon appears green, yellow, or red to indicate the status of the monitor. The colors represent the severity of the problems as defined by the thresholds that you have set. Green signifies that the monitor is running and everything is fine. Yellow is a warning and signifies that the object being monitored is approaching the threshold. Red indicates an alarm and that the object being monitored has reached the threshold value. You can use the predefined monitors that are included with DB2 or you can create your own monitors.

To see what information the Performance Monitor is collecting, click the right mouse button on the object and select **Show Monitor Activity** in the pop-up window.

Use the information from the Performance Monitor to:

- Detect performance problems
- Tune databases for optimum performance
- Analyze performance trends
- Analyze the performance of database applications
- Prevent problems from occurring

The Performance monitor lets you analyze trends by creating a visual presentation of database information such as disk activity, buffer pool usage, amount of prefetch, lock usage, and record blocking at specific intervals.

You use the tool when you need to monitor an existing problem or when you want to observe the performance of your system. It lets you take a snapshot of database activity and performance data at a point in time. These snapshots are used for comparison over time. Each point on the graph represents a data value. The steps for taking snapshots are provided in "Monitoring Performance at a Point in Time" on page 28. This information can help you to identify and analyze potential problems, or identify exception conditions that are based on thresholds that you set. Use the performance tool if you need to know the performance of the database manager and its database applications at a single point in time and look at trends over time. Use it also to get a visual overview of what elements are in a state of alarm. This helps you to identify which parameters may need tuning. You can then look closely at the parameters that have been set for that element and change it to improve performance.

Event Monitor

In contrast to taking a point in time snapshot, an event monitor collects information on database activities over a period of time. This collected information provides a good summary of the activity for a particular database event, for example, a database connection or an SQL statement. Event monitoring records the state of the database at the time specific events occur. It allows you to obtain a trace of the activity on the database. Event monitor records are stored and then analyzed after the data has been captured. Use the event monitor when you need to know how long a transaction took or, for example, how much CPU an SQL statement used. You then use the Event Analyzer to read the data recorded from the event monitor.

For each database connection, there is one connection event record produced. For each statement run in that connect, a statement record is produced. Each connection event record maps to one row in the Connections View window of the Event Analyzer. This window shows information for each application that connected during the monitored period, including:

- Application name
- Execution ID
- Connect time
- Total CPU time
- Lock wait time
- Total sort time
- Deadlocks
- Disconnect time

- Application ID

Each statement event record maps to one row in the Statements View window in the Event Analyzer.

Using the Monitor Tools

The Performance Monitor and the Event Analyzer provide the following benefits:

- Comprehensive, flexible data collection. Over 200 performance variables are supported including buffer pool and I/O, lock and deadlock, sorting, communication, agent, and logging information. Data is shown for database managers, databases, table spaces, tables, buffer pools, connections, transactions, and SQL statements.
- Easy-to-use, intuitive viewing. Data can be viewed in real time using easy-to-read graphs or textual views conveniently organized into logical groups. Both details and summary views are provided, with the ability to access more detailed information.
- Robust alert capabilities. For any performance measurement, you can define exception conditions by specifying a threshold value. The threshold values are used to visually identify when a performance measurement reaches or exceeds the threshold value by plotting a measurement in a particular zone on the performance graph. When the threshold value is reached, you can specify that you want any or all of the following actions to occur:
  - You are notified through the Alert Center.
  - You receive an audible alarm.
  - A program is run.
  - A message is displayed.

  Or, you may decide that no notification should be given.

Figure 2 on page 27 illustrates how the monitors work together.

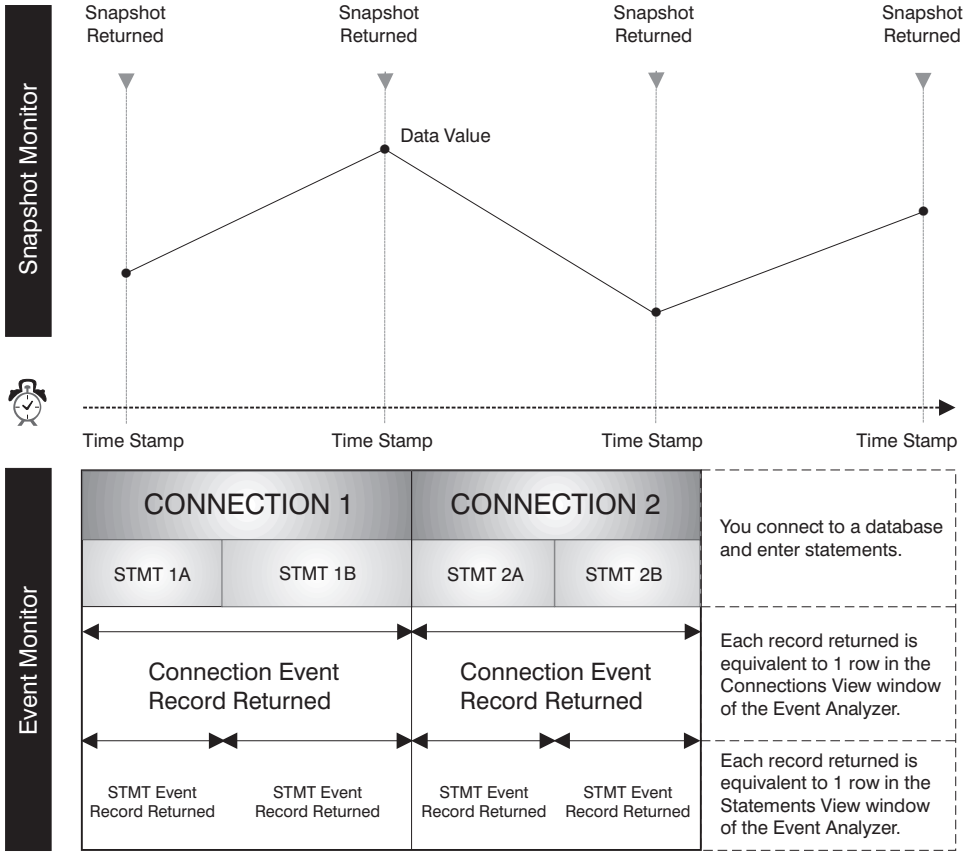


Figure 2. Comparison: Getting Snapshots and Monitoring Events. (Event Monitor, Event Analyzer)

Considerations for Monitoring and Tuning a Database

Before you start monitoring and tuning your database, you should do the following:

- Define your objectives. For example, you may want to understand how applications use resources at the instance level at a specific point in time so that you can, for example, check if database concurrency is reduced when a special application is started. Or you may want to understand which instance-level events occur when an application is running, for example, if there is poor overall performance when a particular application is running.
- Determine what information you will analyze. For example, to see if bottlenecks are hardware related, you may want to take snapshots to monitor database connection activity or table space, buffer pool, and I/O activity. To see if the bottlenecks are environment-related, you would use the Event Analyzer to monitor if:

- Too many database tasks are scheduled during peak time
- There is a high number of user connections
- Database partitioning (hardware load balancing) is not well optimized
- The server is being used for more than just a database server

Some of the visible effects are, for example:

- Queries/responses are slow
- Scheduled tasks are not completing on time
- Applications are timing out

- Decide whether you will use the predefined monitors that are available with DB2 or whether you will create your own monitors.

The next section describes how to take snapshots and how to use the Alert Center to keep track of any performance-related problems.

Monitoring Performance at a Point in Time

If you want to do complex data collection and analyze the data to pinpoint potential problems, use the Performance Monitor to take snapshots of your system and watch performance data change over time.

The tool lets you:

- Graph performance information
- Set the capture frequency of performance snapshots
- View the results of performance calculations
- Define threshold values and threshold actions
- Generate and store alerts
- View summary information (for example, all databases)

The following types of information are captured:

- Information about long-lived activities (such as database activity when an application is taking too long to complete).
- Counters that keep track of information about the current level of activity (such as the number of open cursors for a database).
- Cumulative information about database activity (such as the maximum number of connections made while a database instance is active, or the total number of SQL statements executed against a particular database).

Taking snapshots at predefined intervals provides a picture of the current state of the activity in the database manager and its applications. This information can be used to:

- Detect performance problems
- Analyze performance trends

- Tune database manager and database configuration parameters
- Analyze the performance of database applications

Performance information is available for the following database objects:

- Instances
- Databases
- Tables
- Table spaces
- Database connections

For each, a variety of performance variables can be monitored. The Performance Variable Reference Help, available from the **Help** menu of any Snapshot Monitor window, provides a description of all the performance variables. These variables are organized into categories. The following categories exist:

- Instance: Agents, Connections, Sort
- Database: Lock and Deadlock, Buffer Pool and I/O, Connections, Sort, SQL Statement Activity
- Table: Table
- Table space: Buffer Pool and I/O
- Database Connections: Buffer Pool and I/O, Lock and Deadlock, Sort, SQL Statement Activity

For detailed information on how to generate snapshots, see the online help.

Predefined Monitors

The DB2 Performance Monitor contains a set of predefined monitors, which you can use as they are or which you can copy and modify to meet your requirements. They provide a comprehensive set of performance calculations. You cannot change the name, equation, or text description of an IBM-supplied performance monitor; however, you can change the threshold values and the alert actions. Use the predefined monitors to learn about performance monitoring and to create your own monitors by copying a predefined monitor and adding or removing performance variables from your copy.

The Predefined Monitors that are supplied with DB2 are:

- Monitoring Capacity. Use this monitor to get information on system capacity. This monitor can be checked on a regular basis to see the overall usage of your system over time.
- Sort. Use this monitor to ensure that your sort heap and sort heap threshold parameters are set correctly. This monitor should be run when you first start your system, in peak periods of activity, or as applications change.

- Locking. Use this monitor to determine how much locking is occurring in your system, and whether your lock list parameters are set appropriately.
- Cache. Use this monitor to optimize cache usage. By monitoring these values during peak periods, you can determine if you need to increase the size of the cache.
- Bufferpool. Use this monitor on small tables to determine whether they require their own buffer pools.
- Deadlocks. Use this monitor to determine whether your applications are getting into deadlocks.
- Fast Communication Manager. Use this monitor to see the percentage of memory used to transfer information between nodes.
- Prefetchers. Use this monitor to determine whether you have enough prefetchers defined for the system.
- Disk Performance. Use this monitor to watch input and output. This monitor contains performance variables that focus on disk performance at the database and table space levels.
- Global Memory. Use this monitor to watch application memory usage.
- Long Running Memory. Use this monitor to help determine why a query is taking a long time to complete.
- Gateway Connections. Use this monitor to watch DB2 Connect server connections.

For examples of how to use a predefined monitor, see the online help provided for performance monitoring.

To see a list of available monitors, from the Control Center, click the right mouse button on the **Systems** folder, and select **List Monitors** from the pop-up menu. The List Monitors window opens. It lists the monitors that are stored on the JDBC server to which you are currently connected. For each monitor, you see the name of the monitor, a description, the status, whether it is the default monitor, and who created the monitor. The "Status of the monitors" indicates the status of the monitors on the local system, and not on the JDBC server. The "Default for" level indicates the default monitor at the instance, database, table, table space or connections level. For the predefined monitors, the "Created by" column contains **NULLID**. The right side of the window contains push buttons which allow you to perform various tasks on the monitors. See "Running the Control Center as a Java Applet" on page 46 for more information on the JDBC server.

You can choose which monitor is started as the default monitor for an object.

Once you have started a performance monitor, you can click on the Alert Center button on the toolbar to see the status of any objects that you are

monitoring and which are in a state of alert because they have reached any of their threshold values. They appear only for the period of time during which the threshold is exceeded.

If you want to keep a close watch on the objects being monitored, you can keep the Alert Center open or you can keep the **Show Monitor** window open on the summary page and look for any red or yellow entries. You can also modify the Control Center settings so the Alert Center opens automatically if a new warning or alarm is added to it. From the Alerts Center, you can also temporarily suspend the alerts while monitoring continues.

Action Required When an Object Appears in the Alert Center

You can set the Alert Center to open automatically to display any monitored objects that are in a state of alarm or warning (that is, their thresholds have been exceeded). You can change this default from the Tools Settings window.

If you see an object in the Alert Center, click the right mouse button on the object and select **Performance Monitor —> Show Monitor** to view the performance details for that database object.

Analyzing an Event for a Period of Time

The Event Analyzer is another DB2 performance tool. Use this tool when you want diagnostic information for an event that has taken place. You use the Event Analyzer in conjunction with an event monitor. For example, you can use an event monitor to trace database activity, such as connections, transactions, statements, and deadlocks, while a database is active. An event monitor can also record cumulative performance data that is logged when an application disconnects from the database. After the event monitor has created the event monitor file, you look at your performance information using the Event Analyzer.

The event monitor tools let you perform the following:

- Create event monitors to monitor the types of database events that are of interest to you.
- Activate an event monitor to start collecting event data. The data is stored in a file.
- Stop an event monitor from collecting event data.
- View the trace-type summary information that is produced by the event monitor.
- Remove an event monitor when you no longer have a need for it. You are also given the option to clean up its trace files.
- Display a list of event monitors associated with the database.
- View the definition of an event monitor.

The Event Analyzer lets you view the data generated by an event monitor for the following event types:

- Database connection activity (the period of time between a connection and its disconnection)
- Transactions (units of work)
- SQL statement executions
- Detection of deadlock activity

Event Analyzer

You can create an event monitor for the following event types and then use the Event Analyzer to view the collected information: however, use the `db2evmon` executable (described in the *Command Reference* and the *System Monitor Guide and Reference*) to view data generated for the:

- Deadlocks
- Database activity
- Table space activity
- Table activity
- Statement activity

To analyze event data using an event monitor and the Event Analyzer, follow the steps below. They represent only one example of how to create an event monitor for connection and statement events. To create an event monitor:

1. From a command line in the Command Center, type **db2emcrt**. The Event Monitor window opens.
2. Click on **Event Monitor** and choose Create from the menu. The Create Event Monitor window opens.
3. In the field, specify a name for the event monitor you are creating. This new event monitor cannot have the same name as any existing monitor. Blank spaces are not allowed in the name.
4. In the DB2 Universal Database Enterprise - Extended Edition product only, select a node where the event monitor files will reside from the "On Node" drop down list.
5. In the DB2 Universal Database Enterprise Enterprise - Extended Edition product only, select a scope for the event monitor. By default, the scope is Global.
6. Select one or more of the check boxes to indicate the type of events that you want to monitor. Note that the Deadlocks event type is the default selection.
7. Indicate when you want this monitor to start. Note that "Start now" is the default selection.
8. Define one or more conditions for connections, statements, or transactions that will control monitoring at these levels.

9. Identify a path (directory name) where the monitor will write the event data files.
10. Click on **Options** to open a window for **Specifying Event Monitor File** options. These options determine how monitor output is handled and can affect the performance of your event monitor.
11. Click on OK to create the monitor, or Cancel to exit without creating a monitor.
12. Turn off the event monitoring, by clicking the right mouse button on an event monitor and select **Stop Event Monitoring** from the pop-up menu.

    This forces the event monitor to write the trace file. If the monitor is not turned off, information is only written to disk when the buffer is full or all connections end. From the Event Monitors window, you can view the resultant event data by clicking the right mouse button on the event monitor you created, and selecting **View Event Monitor Files** from the pop-up menu. The Monitored Periods View window opens.

To access the event data from the Event Analyzer:

1. From a command line in the Command Center, type **db2eva** to start the Event Analyzer. The Event Analyzer window opens.
2. In the Path field, identify the path (directory name) where the data files are stored. If the files have not been moved, this will be the path that was specified when the event monitor was created. If the files were moved, then specify that directory. You can click ... to list existing directories.

   Note: If data files are stored remotely, you must FTP the files to your local machine in order to view them. Depending on file size, this transfer could take some time. Files can be transferred to any local path. It is not necessary to choose the same path that was used when they were created.
3. Click **OK** to access the data files contained in the directory, or **Cancel** to exit. The Monitored Periods View window opens.
4. Click the right mouse button on a monitored period, and select **Open as —> Connections** from the pop-up menu. The Connections View window opens. This shows the list of connections that were made during the event monitoring session. (There may be more than one connection listed. The connection you are interested in may not be the first one in the list.)
5. Click the right mouse button on a connection, and select **Open as —> Statements** from the pop-up menu. The SQL Statements View window opens. It displays all statements for the selected connection. Columns of information are provided for each statement, including:
   - **Operation**
   - **Package name**
   - **Creator**

- **Start time**
- **Elapsed time**
- **Total CPU time**
- **Text**

The online help for the event monitor and the Event Analyzer provide detailed instructions for creating event monitors and viewing the resultant event data.

Analyzing SQL Statements

You can view the access plan for explained SQL statements as a graph and use this information to tune your SQL queries for better performance.

An access plan graph shows details of:

- Tables (and their associated columns) and indexes
- Operators (such as table scans, sorts, and joins)
- Table spaces and functions

Prior to Version 6, you would use a tool called Visual Explain to view the access plans. Now, you can no longer invoke Visual Explain as a separate tool from the command line, however, you can still invoke the visual explain *function* from various database objects in the Control Center and from the Command Center. In this section, the term *visual explain function* is used for this capability.

You use the visual explain function to:

- View the statistics that were used at the time of optimization. You can then compare these statistics to the current catalog statistics to help you determine whether rebinding the package might improve performance.
- Determine whether or not an index was used to access a table. If an index was not used, the visual explain function can help you determine which columns might benefit from being indexed.
- View the effects of performing various tuning techniques by comparing the before and after versions of the access plan graph for a query.
- Obtain information about each operation in the access plan, including the total estimated cost and number of rows retrieved (cardinality).

Improving Performance of a Query

You use the visual explain function to analyze and assist in the tuning of SQL statements. It presents a graphical view of the access plan for explained SQL statements. Tables and indexes, and each operation on them, are represented as nodes, and the flow of data is represented by the links between the nodes.

You can use the information available from this graph to find ways to tune your SQL queries for better performance.

The visual explain function captures information about how SQL statements are compiled. This information allows you to understand the plan and potential execution performance of SQL statements.

This information can help you:

- Design application programs.
- Design databases.
- Understand how two tables are joined: the join method that is used, the order in which the tables are joined, the occurrence of sorts, and the type of sorts.
- Determine ways of improving the performance of SQL statements (for example, by creating a new index).
- View the statistics that were used at the time of optimization. You can then compare these statistics to the current catalog statistics to help you determine whether re-binding the package might improve performance. It also helps you determine whether collecting statistics might improve performance.
- Determine whether or not an index was used to access a table. If an index was not used, the visual explain function can help you determine which columns could be included in an index to help improve query performance.
- View the effects of performing various tuning techniques for the purpose of better performance by comparing the before and after versions of the access plan graph for a query.
- Obtain information about each operation in the access plan, including the total estimated cost and number of rows retrieved.

After using the visual explain function to understand the access plan for an explained SQL statement, you may determine that an index might improve the performance of that query. You should use the Index Wizard to receive recommended indexes for the query; or, you could use the RECOMMENDED_INDEXES EXPLAIN-mode. For more information on the Index Wizard, go to the Control Center and enter the Information Center.

Refer to the *Administration Guide: Performance* for more information on the RECOMMENDED_INDEXES EXPLAIN-mode.

Analyzing a Simple Dynamic SQL Statement

This section provides a simple example of how to get started analyzing a dynamic SQL query.

1. From the Control Center, click the right mouse button on the SAMPLE database, and select **Explain SQL** from the pop-up menu. The Explain SQL Statement window opens.
2. In the **SQL text** field, enter the following SQL statement:

   ```
   select * from staff order by name
   ```

3. Click **OK**. The Access Plan Graph window opens. The graph represents the path that the optimizer chose as the most efficient in order to provide the results for your query.
4. Optional: Double-click any of the nodes (for example, the RETURN operator node). The Operator Details window opens, showing the details for that operator.

The explained SQL statement is saved automatically. To view it later:

1. From the Control Center, click the right mouse button on the SAMPLE database, and select **Show explained statements history** from the pop-up menu. The Explained Statements History window opens.
2. Locate the entry you want. You can look at the **SQL text** column to see the SQL statement you had previously explained.
3. Click the right mouse button on the entry, and select **Show access plan** from the pop-up menu. The Access Plan Graph window opens.

The online help for the visual explain function (accessible from the **Help** menu) provides details on how to interpret the Access Plan Graph window in order to improve the performance of SQL statements. The online help also contains detailed examples to help you learn how to use Visual Explain.

Managing Remote Databases

The following section shows you how to:

- Add a remote system
- Add the instance you want to work with for that system
- Add the database you want to work with under that instance

DB2 first checks in the node directory (which contains an entry for all servers to which a database client can connect and the communications protocol used in the connection) to see if the remote system is already known. If the remote system is not known, with a system, instance, or database on a remote system, you need to set yourself up as a client to the remote system.

After you install DB2, you can use the Client Configuration Assistant to search the network for systems, instances, and databases and configure communications for them. You then add the remote system by cataloging it. This creates an entry for the system in the node directory so that its instances and databases can be made known. Next, you must add the instances and

databases for the system by cataloging them to create an entry for them in the node directory and database directory, respectively. This creates an entry for them in the node directory and database directory, respectively). When the configuration is complete, the remote systems are displayed in the Control Center so that you can work with them.

To add a remote system:

1. From the Control Center, click the right mouse button on the **Systems** object and select **Add** from the pop-up menu. The Add System window opens.
2. Enter the system name in the **System name** field.

   If the **Discover** configuration parameter for the instance is set to **search** and the **discover comm** configuration parameter is not blank, you can select **Refresh** to get a list of the remote systems. You can then select one of the systems from the list below the **System name** field.
3. Type the remote instance name in the **Remote instance name** field.
4. Select the type of operating system for the remote system from the **Operating system** list.
5. Select the protocol you want used for communications with the remote locations. For a local system, **Local** is automatically selected and is the only valid protocol. For the remote systems the possible protocols are:
   - APPC
   - IPX/SPX
   - NetBIOS
   - TCP/IP
   - Named pipe (on Windows NT and Windows 9x operating systems only)

   Only the protocols that the computer is currently set up for appear in the listbox.
6. Enter the appropriate protocol parameters.
7. Enter a comment to be associated with the system.
8. Click **Apply** to add the system to the node directory.

Next, add the instance you want to work with on that system:

1. From the Control Center, click the right mouse button on the **Instances** object belonging to the system you just added.
2. Select **Add** from the pop-up menu. The Add Instance window opens.
3. Enter the required values in the fields.
4. Click the **Refresh** push button to have a list of existing instances displayed.
5. Select the instance you want to work with.

6. Click the **Apply** push button, then the **Close** push button.

Finally, add the database you want to work with under that instance:

1. From the Control Center, click the right mouse button on the **Databases** object.
2. Select **Add** from the pop-up menu. The Add Database window opens.
3. Enter the database name, type of communication protocol, and, optionally, an alias. An alias in this case is an alternative name used to identify a database.
4. Click the **Refresh** push button to display a list of existing databases for that instance.
5. Select a database.
6. Click the **Apply** push button, then the **Close** push button.

Managing Users

As a database administrator, you might need to control the type of access people have to data, or restrict their view of the data. The following information tells you how to use the administration tools to manage database authorities and privileges for database objects.

Database *authorities* involve actions on a database as a whole. When a database is created, some authorities are automatically granted to anyone who accesses the database. For example, CONNECT, CREATETAB, BINDADD and IMPLICIT_SCHEMA authorities are granted to all users. Database *privileges* involve actions on specific objects within the database. When a database is created, some privileges are automatically granted to anyone who accesses the database. For example, SELECT privilege is granted on catalog views and EXECUTE and BIND privilege on each successfully bound utility is granted to all users.

Together, privileges and authorities act to control access to an instance and its database objects. Users can access only those objects for which they have the appropriate authorization, that is, the required privilege or authority.

Granting and Revoking Authorities and Privileges

You can use the DB2 administration tools to grant and revoke privileges for users and groups for databases, table spaces, tables, views, and schemas.

1. From the Control Center, click the right mouse button on the database, table, view, schema or index for which you want to grant or revoke privileges. Select **Authorities** or **Privileges** from the pop-up menu. The Authorities window or Privileges window opens.
2. Select the **User** page to work with user authorities or privileges or the **Group** page to work with group authorities or privileges.

3. Select one or more users or groups. To add a user or group to the list, click the **Add User** or **Add Group** push button.
4. Along the bottom of the window, select **Yes**, **No**, or **Grant** for each individual authority or privilege. **Grant** is displayed only for objects for which it is a valid option.
5. When you have finished, click the **Apply** push button.

If you want to review or change the objects that a particular user is authorized to, you can select a user, and click the right mouse button, then add or change authorization to an object or remove authorization.

Moving Data

DB2 provides the Import, Export, and Load utilities to help you move data into a table from existing sources. The information provided in this section is a brief overview of moving data. For more detailed information on moving data, you should refer to the *Data Movement Utilities Guide and Reference* manual.

The Import utility takes data from an input file and inserts it into a table or view. In this case, the input file contains data that was extracted from an existing source of data, such as a Lotus 1–2–3 file or an ASCII file. You can also use the import utility to re-create a table or view that was saved by using the Export utility. The following information tells you how to import data.

Once you have an input file available in a supported format, use the Import notebook to insert data from the file into an existing table. If this table already contains data, you can either replace or append to the existing data with the data in the file.

You can also use the Import notebook to create a new table that is populated by an input file, or delete existing rows in the selected table and repopulate it using data from the input file.

To import a file into an existing table:

1. Open the File page of the Import notebook.
2. Optional. Specify the Import notebook.
3. Optional. Retrieve Large objects.
4. Optional. Specify column import options.
5. Click **OK**

To open the File page of the Import notebook:

1. From the Control Center, expand the object tree until you find the **Tables** folder.

2. Click the **Tables** folder. Any existing tables are displayed in the contents pane.
3. Click the right mouse button on a table in the contents pane and select **Import** from the pop-up menu. The Import notebook opens with the File page displayed.

To specify the file options:

1. In the **Import file** field of the File page, enter the name of the file that contains the data you want to import.
2. Specify the type of file to import by selecting one of the following
   - Non-delimited ASCII format (ASC)

     Non-delimited ASCII data is data that is aligned in columns.
   - Delimited ASCII format (DEL)

     Delimited ASCII data is a commonly used way of storing data that separates column values with a user-defined delimiting character, such as a comma.
   - Worksheet format (WSF)
   - Integrated exchange format (IXF)

     PC/IXF is a structured description of a database table or view. Data that was exported in PC/IXF format can be imported or loaded into another DB2 Universal Database product database.

   See the online help for the specific products and releases that are supported.
3. Optional: Specify file type modifiers by clicking the corresponding **Options** push button. The Options window for that format opens.
4. Select an **Import mode**. The available import modes vary depending on the file type you selected.
5. Optional: In the **Commit records** field, enter the number of records to import before the changes are committed.
6. Optional: In the **Restart** field, enter the number of records in the file to skip before beginning the import action.
7. Optional: In the **Compound** field, type a number to specify how many SQL statements will be executed (in an executable block).
8. Optional: Select the **Insert an implied decimal point on decimal data (IMPLIEDDECIMALPOINT)** check box.
9. In the **Message file** field, type the name of the file that will contain warning and error messages that occur during import.

To retrieve large objects from separate files, use the Large Objects page of the Import notebook to retrieve large objects (LOBs) from the path or paths that store the LOB files:

1. Click the **Retrieve large objects (LOBs) in separate files (LOBSINFILE)** check box to enable the options on the Large Objects page.
2. Specify the location of separate LOB files in the LOB paths list box by clicking the **Add** push button. These paths are searched (in the order in which they appear in the **LOB paths** list box) for the LOB files specified in the LOB column of the input file.
3. Click **OK** to accept the defaults on the other notebook pages and begin the import process.

Specify column import options. Use the **Columns** page of the **Import** notebook to specify column import options:

1. Click one of the radio buttons in the **Include columns by** box to specify the column method that will be used to import data file columns into the table. The available methods vary depending on the file type and mode you selected on the File page.
2. Optional: Specify or change the import file column attributes by clicking the **Change** push button.

   This option is not available if you selected the **Default (method D)** radio button.

Managing Storage

As a database administrator, you need to estimate the size of tables and indexes, to check the amount of space available in a table space, and to add more space to an existing table space when it gets full.

This section describes how to:

- Estimate the size of tables and indexes
- Check the amount of space available in a table space
- Add more space to an existing table space when it starts to get full

Estimating Table and Index Size

You can estimate the amount of storage space required for new or existing tables or indexes by invoking the Estimate Size dialog. Invoke this dialog by selecting individual tables and indexes and clicking the right mouse button on them, or select **Estimate Size** from the Create Table and Create Index windows. The size is estimated on the definition of the particular table and its dependent indexes. The estimate is the projected amount of storage space that would be used when the table has a given number of rows. The minimum and maximum space is also estimated based on the smallest and largest size of variable length fields. When invoked on a table or an index, the **Estimate Size** dialog is prefilled with the specifications of the table, and contains numbers relating to the table and all of its dependent indexes. When you click the **Refresh** push button, the estimated size, minimum size, and maximum

size are updated based on the number you enter in the **New total number of rows** and **New average row length** fields.

Estimating the size of a table or index is helpful if you want to:

- Create a new table and you want to know how large to make the table space.
- Create a new table based on the size estimate of an existing table.
- Know how much space is used by different table and index objects in a table space because the system is running out of storage space.
- Estimate the projected size of a table prior to loading data.

Note: When you use Estimate Size on the DB2 Universal Database Enterprise-Extended Edition product, ensure the size estimates are based on the logical size of the data in the table.

If you have not updated the statistics for the table for some time, you can click the **Run statistics** push button to update the statistics for the selected table. If you select an index and then press the **Run statistics** button, the statistics are run on the related table.

To estimate the size for a table:

- Open the Estimate Size window.
- Select a different value for **New total number of rows** or accept the default.
- Click **Refresh** to view the size estimates for the new value.
- Select a different value for **New average row length** or accept the default.
- Click **Refresh** to view the size estimates for the new value.

Checking Available Space in a Table Space

To check the amount of space available in a DMS table space:

1. From the Control Center, double-click on the **Table Spaces** icon. A list of all the table spaces appears in the contents pane.
2. Scroll to the columns titled **Allocated size**, **Size used** and **Percentage used** to see details related to the amount of space available in a table space. Space is measured in pages where one page is 4 KB.

You can customize the order of the columns and which columns are displayed by using the Customize Columns icon at the bottom of the Contents pane.

To check the available space in an SMS table space, use the facilities provided by your operating system to monitor space usage and to ensure that available room in the directory for the table space is not exhausted.

Adding More Space to a Table Space

Capacity for a DMS table space is the total size of containers allocated to the table space. When a DMS table space reaches capacity (depending on the usage of the table space, 90% is a possible threshold), you should add more space to it. The database manager will automatically rebalance the tables in the DMS table space across all available containers. During rebalancing, data in the table space remains accessible.

For a DMS table space that has reached its capacity, you can add another container:

1. From the Control Center, click the right mouse button on the table space in the Contents pane for which you want to add a container, and select **Alter** from the pop-up menu. The Alter Table Space window opens.
2. Click **Add**. The Add Container window opens.
3. Select the **File** or **Raw device** radio button, and complete the fields. See the online help for assistance.
4. Click **OK**.

In general, you cannot extend the size of an SMS table space very easily because SMS capacity depends on the space available in the file system and the maximum size of the file supported by the operating system. However, depending on your operating system, you may be able to increase the size of a file system using the operating system facilities. For an SMS table space on a UNIX-based system, you can increase the size of the table spaces by using the appropriate UNIX-based system commands. See the documentation for the UNIX-based system you are running. If the file system containing the SMS table space also contains non-DB2 files, you may be able to move these files to another file system, thus making more room available in the file system for DB2's use. You can also perform a redirected restore which involves restoring a table space into a larger number of containers than it was backed up from. You can perform a redirected restore from the Restore Database notebook: From the database you want to restore, select **Restore —> Database** from the pop-up menu.

Troubleshooting

DB2 provides a troubleshooting manual that is intended for technical support representatives for DB2 servers and clients. It helps you to:

- Identify problems or errors in a concise manner
- Solve problems based on their symptoms
- Use available diagnostic tools
- Develop a troubleshooting strategy for your day-to-day DB2 operations.

The *Troubleshooting Guide* presents these basic troubleshooting topics:

- Good troubleshooting practices
- Troubleshooting on the server
- Troubleshooting on the client
- Troubleshooting host communications
- Troubleshooting applications
- Troubleshooting and problem determination.

The *Troubleshooting Guide* presents these advanced troubleshooting topics:

- The DB2 process model
- Using logged information
- Taking traces
- Diagnostic tools for UNIX-based, OS/2, and Microsoft Windows operating systems.

Up-to-date bulletins and technical documentation are available from the World Wide Web at http://www.software.ibm.com/data/db2/library/.

See the section at the end of this book for the details on how to contact IBM.

Replicating Data

Replication is the process of taking changes stored in the database log at the source server and applying them to the target server. You can use replication to define, synchronize, automate, and manage copy operations for data across your enterprise. You can automatically deliver the data from a host system to target sites. For example, you can copy data and applications to branch offices, retail outlets, and even sales representatives' laptops.

The two operational components in replication are Capture and Apply. The Capture component captures changes made to data in source tables which have been defined for replication by reading the database log. The Apply component reads the changed data previously captured and stored in a change data table and applies it to the target tables.

Using the Control Center, you can do the setup required for replication using the **Define as replication source** and **Define subscription** actions. The replication components Capture and Apply run outside the DB2 administration tools.

Replication administrators can perform the following actions from the Control Center:

- Define replication sources
- Define replication subscriptions

- Specify SQL to enhance data during the Apply process

The high-level steps for replicating data are as follows. Refer to the *Replication Guide and Reference* for details.

1. Design a replication scenario (map the source and target tables).
2. Define a replication source (this relates to the Capture action).

To define a replication source:

1. Specify source columns to capture.
2. Choose replication options.
3. Define a replication subscription (this relates to the Apply action).
4. Alter the source table with the Data Capture Changes option.
5. Start Capture to read and store data changes.
6. Start Apply to replicate changes to target tables.

To define a replication subscription:

1. Name the subscription set.
2. Specify the database and target table.
3. Specify the target columns.
4. Specify the row selection.
5. Specify SQL for run-time processing.
6. Set the subscription timing.

Using Lightweight Directory Access Protocol

The Client Configuration Assistant (CCA) can be used to add and delete entries on an LDAP server. All the database instances registered on the LDAP server will be cataloged (cached) automatically on the client. They will show up in Control Center as regular nodes on the navigator tree. These databases can be managed the same way as the other databases that you have cataloged on your machine (except the ADD DATABASE and DROP DATABASE options are not yet available in this release).

To administer an LDAP database, select the database and click the right mouse button. A pop-up window lists the functions which you can perform. For more information on LDAP, see "Appendix J. Lightweight Directory Access Protocol (LDAP) Directory Services" on page 395.

Using a Java Control Center

You can run the Control Center as a Java application or as a Java applet through a web server. In both cases, you need a supported Java Virtual Machine (JVM) installed on your machine to run the Control Center. To run the Control Center as a Java application, you must also have the correct Java Runtime Environment (JRE) installed. A Java Virtual Machine can be a Java Runtime Environment (JRE) for running applications or a Java-enabled browser for running applets.

Java **applications** are run just like other applications on your machine, provided you have the correct JRE installed.

Running the Control Center as a Java Applet

Java **applets** are programs that run within Java-enabled browsers. The Control Center applet code can reside on a remote machine and is served to the client's browser through a web server. If you run the Control Center as a java applet, you must use a supported Java-enabled browser running on a Windows 32-bit or OS/2 operating system. Currently, there are no supported browsers for UNIX operating systems.

The Control Center JDBC Applet Server must be started with a user account that has administrator authority on the machine where the Applet Server resides. You can set your Control Center JDBC Applet Server to start automatically at startup time.

To run the Control Center as a Java applet, you must have a Web server set up on the machine that contains the Control Center applet code and the Control Center JDBC Applet Server. The Web server must allow access to the `sqllib` directory. If you choose to use a virtual directory, substitute this directory for the home directory. For example, if you name your virtual directory `temp`, then you should use `sqllib/temp`. DB2 does not support the installation of the Control Center on a FAT drive for OS/2 because an OS/2 FAT drive does not support long filenames required by Java. For more information on installing and configuring the Control Center as a Java application or Java applet, refer to the *Quick Beginnings* manual for your platform.

Using Your Java Tools for Administration

DB2 includes a set of Java interfaces that allow you to extend the capabilities of the Control Center. The Java interfaces allow you to:

- Add additional items to the menu list when working with objects.
- Add buttons to the Control Center toolbar.

To use this capability, you must have the right level of Java software installed. See the "Appendix K. Extending the Control Center" on page 427 for more information on using this function.

Part 2. Implementing Your Design

Chapter 2. Before Creating a Database

After determining the design of your database, you must create the database and the objects within it. These objects include schemas, nodegroups, table spaces, tables, views, wrappers, servers, nicknames, type mappings, function mappings, aliases, user-defined types (UDTs), user-defined functions (UDFs), triggers, constraints, indexes, and packages. You can create these objects using SQL statements in the command line processor, from the Control Center, or through APIs in applications.

For information on SQL statements, refer to the *SQL Reference* manual. For information on command line processor commands, refer to the *Command Reference*. For information on APIs, refer to the *Administrative API Reference* manual.

Note: Your platform may support a user interface where you can create database objects. This interface can be used instead of the SQL statements, command line processor commands, or APIs. Check the *Quick Beginnings* manual for your platform to determine if you have this capability.

In this chapter the method for completing tasks using the Control Center is highlighted by placing it within a box. This is followed immediately by a comparable method using the command line, sometimes with examples. In some cases, there may be tasks showing only one method. When working with the Control Center, recall that you can use the help there to provide more detail than the overview information found here.

This chapter focuses on the information you should know before you create a database with all of its objects. There are several prerequisite concepts and topics as well as several tasks you must perform before creating a database.

The chapter following this one contains brief discussions of the various objects that may be part of the implementation of your database design.

The final chapter in this part presents topics you must consider before you alter a database and then explains how to alter or drop database objects.

For those areas where DB2 Universal Database interacts with the operating system, some of the topics in this and the following chapters may present operating system-specific differences. You may be able to take advantage of native operating system capabilities or differences beyond those offered by

DB2 UDB. Refer to your appropriate *Quick Beginnings* manuals and specific operating system documentation for precise differences.

As an example, Windows NT** supports an application type known as a "service". DB2 for Windows NT can have a DB2 instance defined as a service. A service can be started automatically at system boot, by a user through the Services control panel applet, or by a Win32-based application that uses the service functions included in the Microsoft** Win32** application programming interface (API). Services can execute even when no user is logged on to the system.

Prerequisites Before Creating a Database

Before you implement a database, you should understand the following prerequisite tasks:

- "Starting DB2"
- "Starting DB2 UDB on Windows NT" on page 53
- "Using Multiple Instances of the Database Manager" on page 53
- "Organizing and Grouping Objects by Schema" on page 54
- "Enabling Parallelism" on page 55
- "Enabling Data Partitioning" on page 57
- "Stopping DB2" on page 59

Starting DB2

You may need to start or stop DB2 during normal business operations; for example, you must start an instance before you can perform the following tasks:

- Connect to a database on the instance
- Precompile an application
- Bind a package to a database
- Access host databases.

To start a DB2 instance on your system:

1. Log in with a user ID or name that has SYSADM, SYSCTRL, or SYSMAINT authority on the instance; or log in as the instance owner.
2. On UNIX operating systems, run the startup script as follows:

```
. INSTHOME/sqllib/db2profile      (for Bourne or Korn shell)
source INSTHOME/sqllib/db2cshrc   (for C shell)
```

   where `INSTHOME` is the home directory of the instance you want to use.
3. Use one of these two methods to start the instance:

a. To start the instance using the Control Center:

1) Expand the object tree until you see the **Instances** folder.
2) Right-click the instance that you want to start, and select **start** from the pop-up menu.

b. To start the instance using the command line, enter:

```
db2start
```

Note: The **db2start** command starts the instance according to the rules in "Setting the Current Instance" on page 68.

Starting DB2 UDB on Windows NT

The `db2start` command will launch DB2 as an NT service. DB2 on Windows NT can still be run as a process by specifying the "/D" switch when invoking DB2START. DB2 can also be started as a service using the Control Panel or "NET START" command.

In order to successfully launch DB2 as a service from DB2START, the user account must have the correct privilege as defined by the Windows NT operating system to start an NT service. The user account can be a member of the Administrators, Server Operators, or Power Users group.

When running in a partitioned database environment, each database partition server is started as an NT service.

Using Multiple Instances of the Database Manager

Multiple instances of the database manager may be created on a single server. This means that you can create several instances of the same product on a physical machine, and have them running concurrently. This provides flexibility in setting up environments.

You may wish to have multiple instances to create the following environments:

- Separate your development environment from your production environment.
- Separately tune each for the specific applications it will service.
- Protect sensitive information from administrators. For example, you may wish to have your payroll database protected on its own instance so that owners of other instances will not be able to see payroll data.

DB2 program files are physically stored in one location on a particular machine. Each instance that is created points back to this location so the program files are not duplicated for each instance created. Several related databases can be located within a single instance.

Instances are cataloged as either local or remote in the node directory. Your default instance is defined by the DB2INSTANCE environment variable. You can *attach* to other instances to perform maintenance and utility tasks that can only be done at an instance level, such as creating a database, forcing off applications, monitoring a database, or updating the database manager configuration. When you attempt to attach to an instance that is not in your default instance, the node directory is used to determine how to communicate with that instance.

The *Command Reference* provides information about the type of connection that is required to execute each command.

DB2 support for multiple instances varies by operating system. Refer to the *Quick Beginnings* guide appropriate to your platform for information on defining multiple DB2 instances on one machine.

To attach to another instance, which may be remote, use the ATTACH command as described in the *Command Reference* manual.

To use the Control Center:

1. Expand the object tree until you see the **Instances** folder.
2. Click on the instance you want to attach.
3. Right-click the selected instance name.
4. In the Attach-DB2 window, type your user ID and password, and click **OK**.

To attach to an instance using the command line, enter:

```
db2 attach to <instance name>
```

For example, to attach you to the instance called `testdb2` that was previously cataloged in the node directory:

```
db2 attach to testdb2
```

After performing maintenance activities for the `testdb2` instance, you can then *detach* from that instance by executing the following command:

```
db2 detach
```

Organizing and Grouping Objects by Schema

Database object names may be made up of a single identifier or they may be *schema-qualified objects* made up of two identifiers. The schema, or high-order part, of a schema-qualified object provides a means to classify or group objects in the database. When an object such as a table, view, alias, distinct type, function, index, package or trigger is created, it is assigned to a schema. This assignment is done either explicitly or implicitly.

Explicit use of the schema occurs when you use the high-order part of a two-part object name when referring to that object in a statement. For example, USER A issues a CREATE TABLE statement in schema C as follows:

```
CREATE TABLE C.X (COL1 INT)
```

Implicit use of the schema occurs when you do not use the high-order part of a two-part object name. When this happens, the CURRENT SCHEMA special register is used to identify the schema name used to complete the high-order part of the object name. The initial value of CURRENT SCHEMA is the authorization ID of the current session user. If you wish to change this during the current session, you can use the SET SCHEMA statement to set the special register to another schema name. Refer to the *SQL Reference* for more information.

As described in "Definition of System Catalog Tables" on page 104, some objects are created within certain schemas when the database is created.

In dynamic SQL statements, a schema qualified object name implicitly uses the CURRENT SCHEMA special register value as the qualifier for unqualified object name references. In static SQL statements, the QUALIFIER precompile/bind option implicitly specifies the qualifier for unqualified database object names.

Before creating your own objects, you need to consider whether you want to create them in your own schema or by using a different schema that logically groups the objects. If you are creating objects that will be shared, using a different schema name can be very beneficial. For more information on how to explicitly create a schema, see "Creating a Schema" on page 116.

Enabling Parallelism

You must modify configuration parameters to take advantage of parallelism within a database partition or within a non-partitioned database. For example, intra-partition parallelism can be used to take advantage of the multiple processors on a symmetric multi-processor (SMP) machine.

Enabling Intra-Partition Parallelism

The Control Center can be used to find out, or modify, the values of individual entries in a specific database, or in the database manager configuration file.

You could also use the GET DATABASE CONFIGURATION and the GET DATABASE MANAGER CONFIGURATION commands to find out the values of individual entries in a specific database, or in the database manager configuration file. To modify individual entries for a specific database or in the database manager configuration file, use the UPDATE DATABASE

CONFIGURATION and the UPDATE DATABASE MANAGER CONFIGURATION commands respectively.

Configuration parameters that affect intra-partition parallelism include the *max_querydegree* and *intra_parallel* database manager parameters, and the *dft_degree* database parameter. For more information on configuration parameters, refer to the *Administration Guide: Performance*.

Enabling Intra-Partition Query Parallelism

In order for intra-partition query parallelism to occur, you must modify database configuration parameters and database manager configuration parameters.

INTRA_PARALLEL
Database manager configuration parameter. Refer to *Administration Guide: Performance* for more information on this parameter.

DFT_DEGREE
Database configuration parameter. Provides the default for the DEGREE bind option and the CURRENT DEGREE special register. Refer to *Administration Guide: Performance* for more information on this parameter.

DEGREE
Precompile or bind option for static SQL. Refer to the *Command Reference* for more information.

CURRENT DEGREE
Special register for dynamic SQL. Refer to the *SQL Reference* for more information.

For more information on the configuration parameter settings, and how to enable applications to process in parallel, refer to ″Configuring DB2″ in the *Administration Guide: Performance*.

Enabling Inter-Partition Query Parallelism

Inter-partition parallelism occurs automatically based on the number of database partitions and the distribution of data across these partitions.

Enabling Utility Parallelism

This section provides an overview of how to enable intra-partition parallelism for the following utilities:

- Load
- Create index
- Backup database / table space
- Restore database / table space

Inter-partition parallelism for utilities occurs automatically based on the number of database partitions.

Load: The Load utility automatically makes use of parallelism, or you can use the following parameters on the LOAD command:

- CPU_PARALLELISM
- DISK_PARALLELISM

Refer to the *Data Movement Utilities Guide and Reference* for information about the LOAD command.

AutoLoader: You can enable multiple split processes for the AutoLoader by specifying the MODIFIED BY ANYORDER parameter for the LOAD specification in the `autoloader.cfg` file. For more information, refer to *Data Movement Utilities Guide and Reference*.

Create Index: To enable parallelism when creating an index:

- The INTRA_PARALLEL database manager configuration parameter must be ON
- The table must be large enough to benefit from parallelism
- Multiple processors must be enabled on an SMP machine.

Refer to the *SQL Reference* for information on the CREATE INDEX statement.

Enabling Data Partitioning

When running in a partitioned database environment, you can create a database from any node that exists in the `db2nodes.cfg` file using the CREATE DATABASE command or the sqlecrea() application programming interface (API). For information, refer to the *Command Reference* and *Administrative API Reference* manuals.

Before creating a partitioned database, you must determine if you will be attaching as a local or remote client to the instance where the database is to be created. Second, you must attach to the instance. You must also select which database partition will be the catalog node for the database. The database partition to which you attach and execute the CREATE DATABASE command becomes the *catalog node* for that particular database.

The catalog node is the database partition on which all system catalog tables are stored. All access to system tables must go through this database partition. All federated database objects (wrappers, servers, nicknames, etc.) are stored in the system catalog tables at this node.

If possible, you should create each database in a separate instance. If this is not possible (that is, you must create more than one database per instance),

you should spread the catalog nodes among the available database partitions. Doing this reduces contention for catalog information at a single database partition.

Note: You should regularly do a backup of the catalog node and avoid putting user data on it (whenever possible), because other data increases the time required for the backup.

When you create a database, it is automatically created across all the database partitions defined in the `db2nodes.cfg` file.

When the first database in the system is created, a system database directory is formed. It is appended with information about any other databases that you create. The system database directory is `sqldbdir` and is located in the `sqllib` directory under your home directory. This directory must reside on a shared file system, (for example, NFS on UNIX platforms) because there is only one system database directory for all the database partitions that make up the partitioned database.

Also resident in the `sqldbdir` directory is the system intention file. It is called `sqldbins`, and ensures that the database partitions remain synchronized. The file must also reside on a shared file system since there is only one directory across all database partitions. The file is shared by all the partitions making up the database.

Configuration parameters have to be modified to take advantage of data partitioning. Use the GET DATABASE CONFIGURATION and the GET DATABASE MANAGER CONFIGURATION commands to find out the values of individual entries in a specific database, or in the database manager configuration file. To modify individual entries in a specific database, or in the database manager configuration file, use the UPDATE DATABASE CONFIGURATION and the UPDATE DATABASE MANAGER CONFIGURATION commands respectively.

The database manager configuration parameters affecting a partitioned database include *conn_elapse*, *fcm_num_anchors*, *fcm_num_buffers*, *fcm_num_connect*, *fcm_num_rqb*, *max_connretries*, *max_coordagents*, *max_time_diff*, *num_poolagents*, and *stop_start_time*.

For more information on configuration parameters, refer to the *Administration Guide: Performance*.

Backup Database / Table Space

To enable I/O parallelism when backing up a database or table space:

- Use more than one target media.
- Configure table spaces for parallel I/O.

- Use the PARALLELISM parameter on the BACKUP command to specify the degree of parallelism.
- Use the WITH `num-buffers` BUFFERS parameter on the BACKUP command to ensure enough buffers are available to accommodate the degree of parallelism. The number of buffers should equal the number of target media you have plus the degree of parallelism selected plus a few extra.

  Also, use a backup buffer size that is:

  – As large as feasible. 4 MB or 8 MB (1024 or 2048 pages) is a good rule of thumb.
  – At least as large as the largest (extentsize * number of containers) product of the table spaces being backed up.

Refer to the *Command Reference* for information on the BACKUP DATABASE command.

Restore Database / Table Space

To enable I/O parallelism when restoring a database or table space:

- Use more than one source media.
- Configure table spaces for parallel I/O.
- Use the PARALLELISM parameter on the RESTORE command to specify the degree of parallelism.
- Use the WITH `num-buffers` BUFFERS parameter on the RESTORE command to ensure enough buffers are available to accommodate the degree of parallelism. The number of buffers should equal the number of target media you have plus the degree of parallelism selected plus a few extra.

  Also, use a restore buffer size that is:

  – As large as feasible. 4 MB or 8 MB (1024 or 2048 pages) is a good rule of thumb.
  – At least as large as the largest (extentsize * number of containers) product of the table spaces being restored.
  – The same as, or an even multiple of, the backup buffer size.

Refer to the *Command Reference* for information on the RESTORE DATABASE command.

Stopping DB2

The **db2stop** command can only be run at the server. No database connections are allowed when running this command; however, if there are any instance attachments, they are forced off before DB2 is stopped.

Note: If command line processor sessions are attached to an instance, you must run the **terminate** command to end each session before running

the **db2stop** command. The **db2stop** command stops the instance defined by the DB2INSTANCE environment variable.

To stop a DB2 instance on your system, you must do the following:

1. Log in or attach to an instance with a user ID or name that has SYSADM, SYSCTRL, or SYSMAINT authority on the instance; or, log in as the instance owner.
2. Display all applications and users that are connected to the specific database that you want to stop. To ensure that no vital or critical applications are running, list applications. You need SYSADM, SYSCTRL, or SYSMAINT authority for this.
3. Force all applications and users off the database. You require SYSADM or SYSCTRL authority to force users.
4. On UNIX operating systems, run the startup script as follows:

   ```
   . INSTHOME/sqllib/db2profile      (for Bourne or Korn shell)
   source INSTHOME/sqllib/db2cshrc   (for C shell)
   ```

   where `INSTHOME` is the home directory of the instance you want to use.
5. Use one of these methods to stop the instance:

   a. Expand the object tree until you find the **Instances** folder.
   b. Click each instance you want to stop.
   c. Right-click any of the selected instances, and select **stop** from the pop-up menu.
   d. On the Confirm stop window, click **OK**.

   To stop the instance using the command line, enter:

   ```
   db2stop
   ```

Details on Creating a Database

Before creating a database, you should consider or carry out the following tasks:

- Designing Logical and Physical Database Characteristics
- Creating an Instance
- Establishing the Environment Variables and the Profile Registry
- Creating a DB2 Administration Server (DAS)
- Creating a Node Configuration File
- Creating the Database Configuration File
- Replicating Configuration Information Using Response Files
- Enabling FCM Communications

Designing Logical and Physical Database Characteristics

You must make logical and physical database design decisions before you create a database. To find out more about logical and physical database design, refer to *Administration Guide: Planning*.

Creating an Instance

An instance is a logical database manager environment where you catalog databases and set configuration parameters. Depending on your needs, you can create more than one instance. You can use multiple instances to do the following:

- Use one instance for a development environment and another instance for a production environment.
- Tune an instance for a particular environment.
- Restrict access to sensitive information.
- Control the assignment of SYSADM, SYSCTRL, and SYSMAINT authority for each instance.
- Optimize the database manager configuration for each instance.
- Limit the impact of an instance failure. In the event of an instance failure, only one instance is affected. Other instances can continue to function normally.

It should be noted that multiple instances have some minor disadvantages:

- Additional system resources (virtual memory and disk space) are required for each instance.
- More administration is required because of the additional instances to manage.

The instance directory stores all information that pertains to a database instance. You cannot change the location of the instance directory once it is created. The directory contains:

- The database manager configuration file
- The system database directory
- The node directory
- The DB2 diagnostic file (`db2diag.log`)
- The node configuration file (`db2nodes.cfg`)
- Any other files that contain debugging information, such as the exception/register dump or the call stack for the DB2 processes.

On UNIX operating systems, the instance directory is located in the `INSTHOME/sqllib` directory, where `INSTHOME` is the home directory of the instance owner.

In a partitioned database system, the instance directory is shared between all database partition servers belonging to the instance. Therefore, the instance directory must be created on a network share drive that all machines in the instance can access.

As part of your installation procedure, you create an initial instance of DB2 called "DB2". On UNIX, the initial instance can be called anything you want within the naming rules guidelines. The instance name is used to set up the directory structure.

To support the immediate use of this instance, the following are set during installation:

- The environment variable DB2INSTANCE is set to "DB2".
- The DB2 registry variable DB2INSTDEF is set to "DB2".

On UNIX, the default can be called anything you want within the naming rules guidelines.

These settings establish "DB2" as the default instance. You can change the instance that is used by default, but first you have to create an additional instance.

Before using DB2, the database environment for each user must be updated so that it can access an instance and run the DB2 programs. This applies to all users (including administrative users).

On UNIX operating systems, sample script files are provided to help you set the database environment. The files are: `db2profile` for Bourne or Korn shell, and `db2cshrc` for C shell. These scripts are located in the `sqllib` subdirectory under the home directory of the instance owner. The instance owner or any user belonging to the instance's SYSADM group can customize the script for all users of an instance. Alternatively, the script can be copied and customized for each user.

The sample script contains statements to:

- Update a user's PATH by adding the following directories to the existing search path: the `bin`, `adm`, and `misc` subdirectories under the `sqllib` subdirectory of the instance owner's home directory.
- Set the DB2INSTANCE environment variable to the instance name.

Setting the DB2 Environment Automatically

Note: This discussion only applies to the UNIX operating system environments.

By default, the scripts affect the user environment for the duration of the current session only. You can change the .profile file to enable it to run the db2profile script automatically when the user logs on using the Bourne or Korn shell. For users of the C shell, you can change the .login file to enable it to run the db2shrc script file.

Add one of the following statements to the .profile or .login script files:

- For users who share one version of the script, add:

```
. INSTHOME/sqllib/db2profile    (for Bourne or Korn shell)
source INSTHOME/sqllib/db2cshrc (for C shell)
```

  where INSTHOME is the home directory of the instance that you wish to use.
- For users who have a customized version of the script in their home directory, add:

```
. USERHOME/db2profile         (for Bourne or Korn shell)
source USERHOME/db2cshrc      (in C shell)
```

  where USERHOME is the home directory of the user.

Setting the DB2 Environment Manually

Note: This discussion only applies to the UNIX operating system environments.

To choose which instance you want to use, enter one of the following statements at a command prompt. The period (.) and the space are required.

- For users who share one version of the script, add:

```
. INSTHOME/sqllib/db2profile    (for Bourne or Korn shell)
source INSTHOME/sqllib/db2cshrc (for C shell)
```

  where INSTHOME is the home directory of the instance that you wish to use.
- For users who have a customized version of the script in their home directory, add:

```
. USERHOME/db2profile         (for Bourne or Korn shell)
source USERHOME/db2cshrc      (in C shell)
```

  where USERHOME is the home directory of the user.

If you want to work with more than one instance at the same time, run the script for each instance that you want to use in separate windows. For example, assume that you have two instances called test and prod, and their home directories are /u/test and /u/prod.

In window 1:

- In Bourne or Korn shell, enter:

```
. /u/test/sqllib/db2profile
```

- In C shell, enter:

```
source /u/test/sqllib/db2cshrc
```

In window 2:

- In Bourne or Korn shell, enter:

```
. /u/prod/sqllib/db2profile
```

- In C shell, enter:

```
source /u/prod/sqllib/db2cshrc
```

Use window 1 to work with the `test` instance and window 2 to work with the `prod` instance.

Note: Enter the **which db2** command to ensure that your search path has been set up correctly. This command returns the absolute path of the DB2 CLP executable. Verify that it is located under the instance's `sqllib` directory.

Multiple Instances on a System

It is possible to have more than one instance on a system. However, you may only work within one instance of DB2 at a time.

The instance owner and the group that is the System Administration (SYSADM) group are associated with every instance. The instance owner and the SYSADM group are assigned during the process of creating the instance. One user ID or username can be used for only one instance. That user ID or username is also referred to as the *instance owner*.

Each instance owner must have a unique home directory. All of the files necessary to run the instance are created in the home directory of the instance owner's user ID or username. If it becomes necessary to remove the instance owner's user ID or username from the system, you could potentially lose files associated with the instance and lose access to data stored in this instance. For this reason, it is recommended that you dedicate an instance owner user ID or username to be used exclusively to run DB2.

The primary group of the instance owner is also important. This primary group automatically becomes the system administration group for the instance and gains SYSADM authority over the instance. Other user IDs or usernames that are members of the primary group of the instance owner also gain this level of authority. For this reason, you may want to assign the instance owner's user ID or username to a primary group that is reserved for the administration of instances. (Also, ensure that you assign a primary group to the instance owner user ID or username; otherwise, the system-default primary group is used.)

If you already have a group that you want to make the system administration group for the instance, you can simply assign this group as the primary group when you create the instance owner user ID or username. To give other users administration authority on the instance, add them to the group that is assigned as the system administration group.

To separate SYSADM authority between instances, ensure that each instance owner user ID or username uses a different primary group. However, if you choose to have a common SYSADM authority over multiple instances, you can use the same primary group for multiple instances.

Add an Instance

If you have Administrative authority on OS/2, or you belong to the Administrative group on Windows NT, or you have root authority on UNIX platforms, you can add additional DB2 instances. The machine where you add the instance becomes the instance-owning machine (node zero). Ensure that you add instances on a machine where an Administration Server resides.

To add another instance, perform the following steps:

1. Log on under a user ID or name that has Administrative authority or belongs to the local Administrators group.
2. To add an instance, use one of the following methods:

   To use the Control Center:

   a. Expand the object tree until you find the **Instances** folder of the system that you want.
   b. Right-click the instance folder, and select **Add** from the pop-up menu.
   c. Complete the information, and click **Apply**.

   To add an instance using the command line, enter:

   ```
   db2icrt <instance_name>
   ```

3. Create an Administration Server.

When using the **db2icrt** command to add another instance of DB2, you should provide the login name of the instance owner and optionally specify the authentication type of the instance. The authentication type applies to all databases created under that instance. The authentication type is a statement of where the authenticating of users will take place. For more information on authentication, see "Chapter 5. Controlling Database Access" on page 221.

Note: You can choose to update your instance configuration using the **db2iupdt** command.

You can change the location of the instance directory from DB2PATH using the DB2INSTPROF environment variable. You require write-access for the instance directory. If you want the directories created in a path other than DB2PATH, you have to set DB2INSTPROF **BEFORE** entering the **db2icrt** command.

DB2 Enterprise - Extended Edition Details When Adding Instances: When working with DB2 Universal Database Enterprise - Extended Edition, you will also need to declare that you are adding a new instance that is a partitioned database system. This is done using `–s eee` on the command line.

UNIX Details When Creating Instances: When working with UNIX operating systems, the **db2icrt** command has the following optional parameters:

- –h or –?

  This parameter is used to display a help menu for the command.
- –d

  This parameter sets the debug mode for use during problem determination.
- –a AuthType

  This parameter specifies the authentication type for the instance. Valid authentication types are SERVER, CLIENT, DCS, or DCE. If not specified, the default is SERVER, if a DB2 server is installed. Otherwise, it is set to CLIENT.

  Notes:

  1. The authentication type of the instance applies to all databases owned by the instance.
  2. On UNIX operating systems, the authentication type DCE is not a valid choice.
- –u FencedID

  This parameter is the user under which the fenced user-defined functions (UDFs) and stored procedures will execute. This is not required if you install the DB2 client or the DB2 Application Development Client. For other DB2 products, this is a required parameter.

  Note: FencedID may not be “root” or “bin”.
- –p PortName

  This parameter specifies the TCP/IP service name or port number to be used. This value will then be set in the instance’s database configuration file for every database in the instance.
- –s InstType

  Allows different types of instances to be added. Valid instance types are: `ee`, `eee` and `client`.

Examples:

- To add an instance for a DB2 server, you can use the following command:

  ```
  db2icrt -u db2fenc1 db2inst1
  ```

- If you installed the DB2 Connect Enterprise Edition only, you can use the instance name as the Fenced ID also:

  ```
  db2icrt -u db2inst1 db2inst1
  ```

- To add an instance for a DB2 client, you can use the following command:

  ```
  db2icrt db2inst1 –s client –u fencedID
  ```

DB2 client instances are created when you want a workstation to connect to other database servers and you have no need for a local database on that workstation.

Windows NT Details When Creating Instances: When working with the Windows NT operating system, the **db2icrt** command has the following optional parameters:

- –s InstType

  Allows different types of instances to be created. Valid instance types are: `ee`, `eee` and `client`.

- /p:InstProf_Path

  This is an optional parameter to specify a different instance profile path. If you do not specify the path, the instance directory is created under the `SQLLIB` directory, and given the shared name `DB2` concatenated to the instance name. Read and write permissions are automatically granted to everyone in the domain. Permissions can be changed to restrict access to the directory.

  If you do specify a different instance profile path, you must create a shared drive or directory. This will allow the opportunity for everyone in the domain to access the instance directory unless permissions have been changed.

- /u:username,password

  When creating a partitioned database environment, you must declare the logon and account name and password of the DB2 service.

- /r:base_port,end_port

  This is an optional parameter to specify the TCP/IP port range for the Fast Communications Manager (FCM). If you specify the TCP/IP port range, you must ensure that the port range is available on all machines in the partition database system.

For example, on DB2 for Windows NT Enterprise - Extended Edition, you could have the following example:

```
db2icrt inst1 –s eee
   /p:\\machineA\db2mpp
   /u:yourname,yourpwd /r:9010,9015
```

Note: The **db2icrt** command grants to the username used to create the instance:

- Act as a part of the operating system
- Create a token object
- Increase quota
- Log on as a service
- Replace a process level token

The instance requires these user rights to access the shared drive, authenticate the user account, and run DB2 as a Windows NT service.

Listing Instances

To get a list of all the instances that are available on a system using the Control Center:

1. Expand the object tree until you find the **Instances** folder for the system.
2. Right-click the Instances folder, and select **Add** from the pop-up menu.
3. On the Add Database window, click **Refresh**.
4. Click the drop-down arrow to see a list of database instances.
5. Click **Cancel** to exit the window.

To list all instances that are available on a system using the command line, enter:

```
db2ilist
```

To determine which instance applies to the current session, enter:

```
set db2instance
```

Note: On UNIX operating systems, enter:

```
db2 get instance
```

Setting the Current Instance

When you run commands to start or stop an instance's database manager, DB2 applies the command to the current instance. DB2 determines the current instance as follows:

- If the DB2INSTANCE environment variable is set for the current session, its value is the current instance. To set the DB2INSTANCE environment variable, enter:

  ```
  set db2instance=<new_instance_name>
  ```

- If the DB2INSTANCE environment variable is not set for the current session, DB2 uses the setting for the DB2INSTANCE environment variable from the system environment variables. On Windows NT, system environment variables are set in System Environment. On Windows 95, they are set in the `autoexec.bat` file. On OS/2, they are set in the `config.sys` file.
- If the DB2INSTANCE environment variable is not set at all, DB2 uses the registry variable, DB2INSTDEF.

  To set the DB2INSTDEF registry variable at the global level of the registry, enter:

  ```
  db2set db2instdef=<new_instance_name> -g
  ```

Auto-Starting Instances

On UNIX operating systems, to enable an instance to auto-start after each system restart, enter the following command:

```
db2iauto -on InstName
```

where `InstName` is the login name of the instance.

On UNIX operating systems, to prevent an instance from auto-starting after each system restart, enter the following command:

```
db2iauto -off InstName
```

where `InstName` is the login name of the instance.

Running Multiple Instances Concurrently

You can start multiple DB2 instances as long as they use the same level of code.

To run multiple instances concurrently using the Control Center:

1. Expand the object tree until you find the **Databases** folder.
2. Right-click an instance, and select **Start** from the pop-up menu.
3. Repeat Step 2 until you have started all the instances that you want to run concurrently.

To run multiple instances concurrently using the command line:

1. Set the DB2INSTANCE variable to the name of the other instance that you want to start by entering:

   ```
   set db2instance=<another_instName>
   ```

2. Start the instance by entering the **db2start** command.

License Management

The management of licenses for your DB2 products is done primarily through the License Center within the Control Center of the online interface to the product. From the License Center you can check the license information, statistics, registered users, and current users for each of the installed products.

Establishing the Environment Variables and the Profile Registry

Environment and registry variables control your database environment.

Prior to the introduction of the DB2 profile registry, changing your environment variables on Windows or OS/2 workstations (for example) required you to change an environment variable and reboot. Now, your environment is controlled, with a few exceptions, by registry variables stored in the DB2 profile registries. Users with system administration (SYSADM) authority for a given instance can update registry values for that instance. Use the **db2set** command to update registry variables without rebooting; this information is stored immediately in the profile registries. The DB2 registry applies the updated information to DB2 server instances and DB2 applications started after the changes are made.

When updating the registry, changes do not affect the currently running DB2 applications or users. Applications started following the update use the new values.

Note: The DB2 environment variables `DB2INSTANCE`, `DB2NODE`, `DB2PATH`, and `DB2INSTPROF` may not, depending on the operating system, be stored in the DB2 profile registries. In order to update these environment variables, the **set** command must be used. These changes are in effect until the next time the system is rebooted. On UNIX platforms, the **export** command may be used instead of the **set** command.

Using the profile registry allows for centralized control of the environment variables. "DB2 Registry and Environment Variables" in the *Administration Guide: Performance* lists many of the environment variables and registry variables. Different levels of support are now provided through the different environment profiles. Remote administration of the environment variables is also available when using the DB2 Administration Server.

There are four profile registries:

- The DB2 Instance Level Profile Registry. The majority of the DB2 environment variables are placed within this registry. The environment variable settings for a particular instance are kept in this registry. Values defined in this level override their settings in the global level.
- The DB2 Global Level Profile Registry. If an environment variable is not set for a particular instance, this registry is used. This registry has the

machine-wide environment variable settings. In DB2 UDB EEE, one global-level profile exists at each machine.
- The DB2 Instance Node Level Profile Registry. In a system where the database is divided across different database partitions, this registry resides on every node (that is, machine), and contains environment variable settings for all instances storing data on the node. Values defined at this level override comparable settings in the instance and global levels.
- The DB2 Instance Profile Registry. This registry contains a list of all instance names recognized by this system.

Users can override DB2 Instance Profile Registry environment variable settings for their session by changing session environment variable settings using the **set** command (or the export command on UNIX platforms).

DB2 configures the operating environment by checking for registry values and environment variables and resolving them in the following order:
1. Environment variables set with the `set` command. (Or the `export` command on UNIX platforms.)
2. Registry values set with the instance node level profile (using the `db2set -i` command with a node number as shown below).
3. Registry values set with the instance profile (using the `db2set -i` command as shown below).
4. Registry values set with the global profile (using the `db2set -g` command as shown below).

Using the db2set Command

The **db2set** command supports the local declaration of the registry variables (and environment variables).

To display help information for the command, use:

```
db2set ?
```

To list the complete set of all supported registry variables, use:

```
db2set -lr
```

To list all defined registry variables for the current or default instance, use:

```
db2set
```

To list all defined registry variables in the profile registry, use:

```
db2set -all
```

To show the value of a registry variable in the current or default instance, use:

```
db2set registry_variable_name
```

To show the value of a registry variable at all levels, use:

```
db2set registry_variable_name -all
```

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

```
db2set registry_variable_name= -i instance_name
node_number
```

To delete a variable's value and to restrict its use, if it is defined at a higher profile level, enter:

```
db2set registry_variable_name= -null instance_name
```

This command will delete the setting for the parameter you specify and restrict high level profiles from changing this variable's value (in this case, DB2 global-level profile). However, the variable you specify could still be set by a lower level profile (in this case, the DB2 node-level profile).

To change a registry variable for in the current or default instance, use:

```
db2set registry_variable_name=new_value
```

To change a registry variable default for all databases in the instance, use:

```
db2set registry_variable_name=new_value
   -i instance_name
```

To change a registry variable default for all instances in the system, use:

```
db2set registry_variable_name=new_value -g
```

To set registry variables at the user level, use:

```
db2set -ul
```

To set registry variables at the user level for a specific user, use:

```
db2set -ul user_name
```

Notes:

1. The parameters "-i", "-g", and "-ul" cannot be used at the same time in the same command.
2. Some parameters will always default to the global level profile. They cannot be set at the instance or node level profiles; for example, `db2system` and `db2instdef`.
3. On UNIX, you must have system administration (SYSADM) authority to change registry values for an instance. Only users with root authority can change parameters in global-level registries.

When running in an LDAP environment, it is possible to set a DB2 registry variable value in LDAP such that its scope is global to all machines and all

users that belong to a directory partition or to a Windows NT domain. Currently, the only DB2 registry variable that can be set at the LDAP global level is DB2LDAP_SEARCH_SCOPE.

To set this variable at the LDAP global level, use the `-gl` option for the `db2set` command.

Note: This is different from the `-g` option which is used to set DB2 registry variables at the machine global level. `-gl` is specific to the LDAP global level. Also, setting this DB2 registry variable in LDAP is only supported on Windows platforms.

To set the search scope value at the global level in LDAP, use:

```
db2set -gl db2ldap_search_scope = value
```

where the *value* can be "local", "domain", or "global".

To change a registry variable default for a particular node in an instance, use:

```
db2set registry_variable_name=new_value
   -i instance_name node_number
```

To reset a registry variable for an instance back to the default found in the Global Profile Registry, use:

```
db2set -r registry_variable_name
```

To reset a registry variable for a node in an instance back to the default found in the Global Profile Registry, use:

```
db2set -r registry_variable_name node_number
```

Setting Environment Variables on OS/2

It is strongly recommended that all DB2-specific registry variables be defined in the DB2 profile registry. If DB2 variables are set outside of the registry, remote administration of those variables is not possible, and the workstation must be rebooted in order for the variable values to take effect.

On OS/2, you should have no environment variables defined in config.sys apart from DB2PATH and DB2INSTPROF. All variables should be defined in the profile registries using the **db2set** command except for those that remain true environment variables.

DB2INSTANCE also remains a true environment variable, however, it is not required if you make use of the DB2INSTDEF registry variable. This registry variable defines the default instance name to use if DB2INSTANCE is not set.

DB2INSTANCE and DB2PATH are set when DB2 is installed; DB2INSTPROF can be set after installation. The environment variable DB2PATH must be set;

this environment variable is set during installation and you should not modify it. Setting DB2INSTANCE and DB2INSTPROF environment variables is optional.

To determine the setting of an environment variable, enter:

```
set variable
```

To change the setting of an environment variable, enter the following command:

```
set variable=value
```

To set system environment variables, do the following: Edit the config.sys file, and reboot the system to have the change take effect.

The different profile registries are located according to the following:

- The DB2 Instance Level Profile Registry file is located under:

  ```
  %DB2INSTPROF%\instance_name\PROFILE.ENV
  ```

- The DB2 Global Level Profile Registry is located under:

  ```
  %DB2INSTPROF%\DEFAULT.ENV
  ```

- The DB2 Instance Profile Registry is located under:

  ```
  %DB2INSTPROF%\PROFILES.REG
  ```

Setting Environment Variables on Windows NT and Windows 95

It is strongly recommended that all DB2-specific registry variables be defined in the DB2 profile registry. If DB2 variables are set outside of the registry, remote administration of those variables is not possible, and the workstation must be rebooted in order for the variable values to take effect.

Windows 32-bit operating systems have one system environment variable, DB2INSTANCE, that can only be set outside the profile registry; however, you are not required to set DB2INSTANCE. The DB2 profile registry variable DB2INSTDEF may be set in the global level profile to specify the instance name to use if DB2INSTANCE is not defined.

DB2 Enterprise - Extended Edition servers on Windows NT have two system environment variables, DB2INSTANCE and DB2NODE, that can only be set outside the profile registry. You are not required to set DB2INSTANCE. The DB2 profile registry variable DB2INSTDEF may be set in the global level profile to specify the instance name to use if DB2INSTANCE is not defined.

The DB2NODE environment variable is used to route requests to a target logical node within a machine. This environment variable must be set in the session in which the application or command is issued and not in the DB2 profile registry. If this variable is not set, the target logical node defaults to the logical node which is defined with port zero (0) on the machine.

To determine the settings of an environment variable, use the **echo** command. For example, to check the value of the DB2PATH environment variable, enter:

```
echo %db2path%
```

To set system environment variables, do the following:

On Windows 95 and Windows 98: Edit the *autoexec.bat* file, and reboot the system to have the change take effect.

On Windows NT 4.x: You can set the DB2 environment variables DB2INSTANCE, DB2PATH, and DB2INSTPROF as follows:

- Select **Start**, **Settings**, **Control Panel**.
- Double-click on the **System** icon.
- In the System Control Panel, in the System Environment Variables section, do the following:
  1. If the DB2INSTANCE variable does not exist:
     a. Select any system environment variable.
     b. Change the name in the *Variable* field to DB2INSTANCE.
     c. Change the *Value* field to the instance name, for example db2inst.
  2. If the DB2INSTANCE variable already exists, append a new value:
     a. Select the DB2INSTANCE environment variable.
     b. Change the *Value* field to the instance name, for example db2inst.
  3. Select Set.
  4. Select OK.
  5. Reboot your system for these changes to take effect.

Note: The environment variable DB2INSTANCE can also be set at the session (process) level. For example, if you want to start a second DB2 instance called TEST, issue the following commands in a command window:

```
set db2instance=TEST
db2start
```

The profile registries are located as follows:

- The DB2 Instance Level Profile Registry in the Windows NT operating system registry, with the path:

  ```
  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES\instance_name
  ```

  Note: The *instance_name* is specific to the database partition you are working with.
- The DB2 Global Level Profile Registry in the Windows NT registry, with the path:

  ```
  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\GLOBAL_PROFILE
  ```

- The DB2 Instance Node Level Profile Registry in the Windows NT registry, with the path:

  ...\SOFTWARE\IBM\DB2\PROFILES*instance_name*\NODES*node_number*

  Note: The *instance_name* and the *node_number* are specific to the database partition you are working with.

DB2 UDB provides the capability of accessing DB2 UDB registry variables at the instance level on a remote machine. Currently, DB2 UDB registry variables are stored in three different levels: machine or global level, instance level, and node level. The registry variables stored at the instance level (including the node level) can be redirected to another machine by using DB2REMOTEPREG. When DB2REMOTEPREG is set, DB2 UDB will access the DB2 UDB registry variables from the machine pointed to by DB2REMOTEPREG. For example,

```
db2set DB2REMOTEPREG=rmtwkstn
```

where *rmtwkstn* is the remote workstation name.

Note: Care should be taken in setting this option since all DB2 instance profiles and instance listings will be located on the specified remote machine name.

This feature may be used in combination with setting DBINSTPROF to point to a remote LAN drive on the same machine that contains the registry.

Setting Environment Variables on UNIX Systems

It is strongly recommended that all DB2-specific registry variables be defined in the DB2 profile registry. If DB2 variables are set outside of the registry, remote administration of those variables is not possible.

On UNIX operating systems, you must set the system environment variable DB2INSTANCE.

The scripts **db2profile** (for Korn shell) and **db2cshrc** (for Bourne shell or C shell) are provided as examples to help you set up the database environment. You can find these files in `insthome/sqllib`, where `insthome` is the home directory of the instance owner.

These scripts include statements to:

- Update a user's path with the following directories:
  - `insthome/sqllib/bin`
  - `insthome/sqllib/adm`
  - `insthome/sqllib/misc`
- Set `DB2INSTANCE` to the default local `instance_name` for execution.

Note: Except for PATH and DB2INSTANCE, all other DB2-supported variables must be set in the DB2 profile registry. To set variables that are not supported by DB2, define them in your script files, **db2profile** and **db2cshrc**.

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

To change the environment variable for the current session, issue commands similar to the following:

- For Korn shell:

```
db2instance=inst1
export db2instance
```

- For Bourne shell or C shell:

```
set db2instance inst1
```

In order for the DB2 profile registry to be administered properly, the following file ownership rules must be followed on UNIX operating systems. (For information on DB2 Administration Server (DAS), see "Creating a DB2 Administration Server (DAS)" on page 78.)

- The DB2 Instance Level Profile Registry file is located under:

```
INSTHOME/sqllib/profile.env
```

  The access permissions and ownership of this file should be:

```
-rw-r--r-- Instance_Owner DAS_Instance_Group profile.env
```

  The *INSTHOME* is the home path of the instance owner.

- The DB2 Global Level Profile Registry is located under:
  - */var/db2/<version_id>/default.env* for AIX, Solaris, SINIX, and NUMA-Q(Sequent) operating systems (where <version_id> is the current version).
  - */var/opt/db2/<version_id>/default.env* for the HP-UX operating system (where <version_id> is the current version).

  The access permissions and ownership of this file should be:

```
-rw-r--r-- DAS_Instance_Owner DAS_Instance_Group default.env
```

  In order to modify a global registry variables, a user must be logged on as: root or the DAS instance owner. See "Creating a DB2 Administration Server (DAS)" on page 78 for more information on the DB2 Administration Server.

- The DB2 Instance Node Level Profile Registry is located under:

```
INSTHOME/sqllib/nodes/node_number.env
```

The access permissions and ownership of the directory and this file should be:

```
drwxrwxr-x Instance_Owner DAS_Instance_Group nodes

-rw-r--r-- Instance_Owner DAS_Instance_Group node_number.env
```

Note: The *Instance_Owner* and the *DAS_Instance_Owner* should both be members of the *DAS_Instance_Group*.

The *INSTHOME* is the home path of the instance owner.

- The DB2 Instance Profile Registry is located under:
  - `/var/db2/<version_id>/profiles.reg` for AIX, Solaris, SINIX, and NUMA-Q(Sequent) operating systems (where <version_id> is the current version).
  - `/var/opt/db2/<version_id>/profiles.reg` for the HP-UX operating system (where <version_id> is the current version).

  The access permissions and ownership of this file should be:

  ```
  -rw-r--r-- root system profiles.reg
  ```

Creating a DB2 Administration Server (DAS)

DB2 Administration Server (DAS) is a special DB2 administration control point used only to assist with administration tasks on other DB2 servers. You must have a running DAS if you want to use the Client Configuration Assistant or the Control Center. DAS assists the Control Center and Client 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 execution of both DB2 and operating system command scripts. These command scripts are user-defined. The Control Center is used to define the schedule of jobs, view the results of completed jobs, and perform other administrative tasks against jobs located either remotely or locally to the DAS.
- 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 Client 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 host system on behalf of a client request from the Control Center or the Client Configuration Assistant.

Authorized access to DAS requires clients with SYSADM authority. All of the clients can be part of the SYSADM_GROUP configuration parameter.

Some of the requested tasks may require specific authority to run. The DAS runs under the identifier of a specific user. The privileges granted to that user must be restricted to only those commands associated with the tasks or operations to be carried out by the administrator. Generally, the tasks or operations required include:

- Query the operating system (OS) configuration information.
- Query the OS for user and group information.
- Act against other DB2 instances to start or stop them.
- Execute scheduled jobs.
- Collect information for Connectivity and Protocol Configuration.

For more information on setting up DAS communications, refer to the *Quick Beginnings* for your platform.

Creating the DAS

Typically, the setup program creates a DAS on the instance-owning machine during DB2 installation. If, however, the setup program failed to create it, you can manually create a DAS.

As an overview of what occurs during the installation process as it relates to DAS, consider the following:

- On the OS/2 or Windows NT platforms:

  Log on to the machine you want to create the DAS on using an account that has local Administrator authority. If a specific user is to be identified, create a user with local Administrator authority. Enter `db2admin create`. If a specific user account is desired, you must use "/USER:" and "/PASSWORD:" when issuing `db2admin create`.)

  When creating the DAS, you can optionally provide a user account name and a user password. If valid, the user account name and password will identify the owner of the DAS. Do not use the user ID or account name created for the DAS as a User Account. Set the password for the account name to "Password Never Expires". After you create the DAS, you can establish or modify its ownership by providing a user account name and user password with the **db2admin setid** command. Refer to the *Command Reference* for more information on this command.

  On DB2 UDB for Windows NT Enterprise - Extended Edition, if you are using the Client Configuration Assistant or the Control Center to automate connection configuration to a DB2 server, the database partition server that is on the same machine as the DAS will be the coordinator node. This means that all physical connections from the client to the database will be

directed to the database partition server on the instance-owning machine before being routed to other database partition servers.

On DB2 UDB for Windows NT Enterprise - Extended Edition, creating additional Administration Servers on other machines allows the Client Configuration Assistant or Control Center to configure other systems as coordinator nodes using DB2 Discovery. To do this, perform the following:

1. Log on to the machine using an account that has local Administrator authority.
2. Create a Windows NT account that has local Administrator authority that will be used by the DAS. Ensure that the username of the account adheres to the DB2 naming conventions. When creating the account for the DAS, note the following:
   - Do not use the account for the DAS as a User Account.
   - Set the password for the account to `Password Never Expires`.
3. Run the following command:

   ```
   db2admin create /user:username
      /password:passwrd
   ```

   where `username` and `passwrd` are the username and password for the DAS.

- On UNIX platforms:
  1. Ensure that you have root authority.
  2. At a command prompt, issue the following command from the `instance` subdirectory in the path of the DB2 Universal Database instance:

     ```
     dasicrt ASName
     ```

     - On AIX:

       ```
       /usr/lpp/db2_nn_00&/instance/
          dasicrt ASname
       ```

     - On HP-UX, NUMA-Q(Sequent), or Solaris:

       ```
       /opt/IBMdb2/<version_id>/instance/
          dasicrt ASname
       ```

     - On Linux:

       ```
       /usr/IBMdb2/<version_id>/instance/
          dasicrt ASname
       ```

     where *ASName* is the instance name of the Administration Server and db2_nn_00& or <version_id> is the current version identifier.

     Note: If you are running NIS and NIS+, you need to set up the user and group names in such a way that:
     - The primary group of the DAS must be in the secondary group of all the instances.

– The secondary group of the DAS must contain the primary group of all the instances.

Secondary group lists are modified automatically only if NIS and NIS+ is not running on the system.

Because a user ID can only own one instance, you must have a separate user ID to own each DB2 Administration Server (DAS) that you create.

Once you create an Administration Server, you should use it to establish directory structures and access permissions.

Starting and Stopping the DAS

To manually start or stop the DAS, you must first log on to the machine using an account or user ID that has local administrative authority.

When working on DB2 for OS/2 or DB2 for Windows NT, you must do the following:

- To start the DAS, enter `db2admin start`
- To stop the DAS, enter `db2admin stop`

Note: For both cases under Windows NT, the person using these commands must have SYSADM, SYSCTRL, or SYSMAINT authority.

When working on DB2 for any of the UNIX operating systems, you must do the following:

- To start the DAS:
  1. Log in as the DAS owner.
  2. Run the start up script using one of the following:

     ```
     . INSTHOME/sqllib/db2profile    (for Bourne or Korn shell)
     source INSTHOME/sqllib/db2cshrc (for C shell)
     ```

     where `INSTHOME` is the home directory of the instance.
  3. To start the DAS use the **db2admin** command:

     ```
     db2admin start
     ```

  Note: The DAS is automatically started after each system reboot.
- To stop the DAS:
  1. Log in as the DAS owner.
  2. Run the start up script using one of the following:

     ```
     . INSTHOME/sqllib/db2profile    (for Bourne or Korn shell)
     source INSTHOME/sqllib/db2cshrc (for C shell)
     ```

     where `INSTHOME` is the home directory of the instance.

3. Stop the DAS using the **db2admin** command as follows:

```
db2admin stop
```

Note: For both cases under UNIX, the person using these commands must have logged on with the authorization ID of the DAS owner.

Listing the DAS

To obtain the name of the DAS instance on your machine, enter:

```
db2admin
```

This command is also used to start or stop the DAS, create a new user and password, drop a DAS instance, and establish or modify the user account associated with the DAS instance.

Configuring the DAS

To see the current values for those administration configuration parameters relevant to the DAS, enter:

```
db2 get admin cfg
```

This will show you the current values that were given as defaults during the installation of the product or those that were given during previous updates to the configuration parameters.

To update individual entries in the database manager configuration file relevant to the DAS, enter:

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

Refer to the *Command Reference* for more information on which database manager configuration parameters can be modified.

To reset the configuration parameters to the recommended database manager defaults, enter:

```
db2 reset admin cfg
```

Changes to the database manager configuration file become effective only after they are loaded into memory (that is, when a `db2admin stop` is followed by a `db2admin start`; or, in the case of a Windows NT platform, stopping and starting the service.)

To set up the communications protocols for the DAS, refer to the *Quick Beginnings* for your platform.

Security Considerations for the DAS

You must first log on to the machine using an account or user ID that has local Administrator authority.

Note: On Windows NT, you should not use the **Services** utility in the **Control Panel** to change the logon account for the DAS since some of the required access rights will not be set for the logon account. Always use the **db2admin** command to set or change the logon account for the DB2 Administration Server (DAS).

After creating the DAS, you can set or change the logon account using the **db2admin** command as follows:

```
db2admin setid username password
```

where *username* and *password* are the username and password of an account that has local Administrator authority.

It is recommended that the user ID or the username have SYSADM authority on each of the servers within the environment so that it can start or stop other instances if required.

Updating the DAS

On UNIX operating systems, if DB2 is updated by installing a Program Temporary Fix (PTF) or a code patch, all DB2 Administration Servers (DAS) as well as all existing instances should be updated. To update a DAS, use the **dasiupdt** command available in the `instance` subdirectory under the subdirectory specific to the installed DB2 version and release.

You must first log on to the machine as "root" (on UNIX), using an account, or with a user ID that has local administrative authority.

The command is used as follows:

```
dasiupdt InstName
```

The `InstName` is the login name of the instance owner. There are also optional parameters for this command that can be placed before the `InstName` and separated by spaces:

- –h or –?

  Displays a help menu for this command.
- –d

  Sets the debug mode, which is used for problem analysis.

Removing the DAS

You must first log on to the machine as "root" (on UNIX), using an account, or with a user ID that has local administrative authority.

To remove the DAS:

- On the OS/2 or Windows NT operating systems:

1. Stop the DAS, using `db2admin stop`.
2. Backup (if needed) all the files in the `db2das00` subdirectory under the `sqllib` subdirectory. The instance directory is indicated by the *DB2INSTPROF* registry variable.

   Note: This example assumes `db2das00` is the name of the DAS to be removed.
3. Drop the DAS, using `db2admin drop`.

Note: Under Windows NT, the person using this command must have SYSADM, SYSCTRL, or SYSMAINT authority.

- On UNIX operating systems:
  1. Log in as the DAS owner.
  2. Run the startup script using one of the following:

     ```
     . INSTHOME/sqllib/db2profile    (for Bourne or Korn shell)
     source INSTHOME/sqllib/db2cshrc (for C shell)
     ```

     where `INSTHOME` is the home directory of the instance.
  3. Stop the DAS using the **db2admin** command as follows:

     ```
     db2admin stop
     ```
  4. Back up (if needed) all the files in the `sqllib` subdirectory under the home directory of the DAS. The instance directory is indicated by the *DB2INSTPROF* registry variable.
  5. Log off.
  6. Log in as `root` and remove the DAS using the **dasidrop** command as follows:

     ```
     dasidrop ASName
     ```

     where the `ASName` is the instance name of the Administration Server. This command is found in the `instance` subdirectory under the subdirectory specific to the installed DB2 version and release.

     Note: The **dasidrop** command removes the `sqllib` directory under the home directory of the DB2 Administration Server (DAS).

Setting Up DAS with EEE Systems

The following information shows the steps necessary to configure DB2 EEE servers (Solaris, NT, Sequent, HP-UX, and AIX) for remote administration using the Control Center.

During installation, the setup program creates a single DAS on the instance-owning machine. You may want to create additional DAS on other machines to allow the Control Center or the Client Configuration Assistant

access to other coordinator nodes. The overhead of working as a coordinator node can then be spread to more than one node in an instance.

To distribute the coordinator function:

1. Create a new DAS on the selected additional machines in the partitioned database system.
2. Catalog each DAS as a separate system in the Control Center or Client Configuration Assistant.
3. Catalog the same instance under each new system, and each time specify the same machine name used to catalog the DAS.

There are two aspects to configuration: That which is required for the DB2 Administration Server (DAS), and that which is recommended for the target, administered DB2 instance. In the three sections which follow, a section is devoted to each of the two configuration topics. Each of the configuration topics is preceded by a section describing the assumed environment.

Example Environment:

product/version:
DB2 UDB EEE V7.1

install path:
install_path

TCP services file:
tcp_services_file

DB2 Instance:

name: db2inst

owner ID:
db2inst

instance path:
instance_path

nodes: 3 nodes, db2nodes.cfg:

- 0 hostA 0 hostA0switch
- 1 hostA 1 hostA1switch
- 2 hostB 0 hostBswitch

DB name:
db2instDB

DAS:

name: db2as

owner/user ID:
db2as

instance path:
das_path

install/run host:
hostA

internode communications port:
16000 (unused port for hostA and hostB)

Note: Please substitute site-specific values for the above fields. For example, the following table contains example pathnames for each supported EEE platform:

Table 1. Example Pathnames for Supported EEE Platforms

Paths	DB2 UDB EEE for AIX	DB2 UDB EEE for Solaris	DB2 UDB EEE for Windows NT
install_path	/usr/lpp/<v_r_ID>	/opt/IBMdb2/<v_r_ID>	C:\sqllib
instance_path	/home/db2inst/sqllib	/home/db2inst/sqllib	C:\profiles\db2inst
das_path	/home/db2as/sqllib	/home/db2as/sqllib	C:\profiles\db2as
tcp_services_file	/etc/services	/etc/services	C:\winnt\system32 \drivers\etc\services

In the table, <v_r_ID> is the platform-specific version and release identifier. For example in DB2 UDB EEE for AIX in Version 5.2, the <v_r_ID> is db2_05_00.

When installing DB2 UDB EEE, the setup program creates a DAS on the instance-owning machine. The database partition server resides on the same machine as the DAS and is the connection point for the instance. That is, this database partition server is the coordinator node for requests issued to the instance from the Control Center or the Client Configuration Assistant.

DAS Configuration: The DAS is an administrative control point which performs certain tasks on behalf of the Control Center. There can be at most one DAS per physical machine. In the case of an EEE instance which consists of several machines, at least one of the machines must be running a DAS so that the Control Center can administer the EEE instance. This DAS (db2as) "represents" the system that is present in the Control Center navigator tree as the parent of the target DB2 instance (db2inst).

For example, db2inst consists of three nodes distributed across two physical machines or hosts. The minimum requirement can be fulfilled by running db2das on either hostA **or** hostB.

Notes:

1. The number of partitions present on hostA does not have any bearing on the number of DASes that can be run on that host. You can run only one copy of db2as on hostA regardless of the multiple logical nodes (MLN) configuration for that host.
2. It is not necessary to create the DAS ID, db2as, on all hosts. Rather, it is necessary for it to exist only on the host upon which it is running. As well, it is not necessary for the home directory of the DAS ID to be mounted on all hosts. In particular with this example, the ID db2as must exist on hostA, is not required on hostB, and db2as's home directory does not need to be mounted on hostB.

Control Center Communications with DAS: Service Ports: The Control Center communicates with the DAS using a TCP service port, 523. Since this port is reserved for exclusive use by DB2 UDB, it is not necessary to insert new entries into the *tcp_services_file*.

Internode Administrative Communications: Service Ports: For some administrative tasks, the DAS must establish communications with all nodes. In order to do so, a named TCP port must be defined in the *tcp_services_file* for each host which participates in the instance.

Note: Windows NT EEE will attempt to add the TCP port entry into the *tcp_services_file* for you.

For example, db2inst is defined across two hosts, hostA and hostB. As specified in "Example Environment" on page 85, port 16000 is unused on both hosts. Therefore, the following line must be inserted into the *tcp_services_file* for both hostA and hostB.

```
db2ccmsrv  16000/tcp
```

The `db2ccmsrv` port name must be present, spelled exactly as shown above, and the same port number selected must be used on all hosts.

Internode Administrative Communications: UNIX DB2 EEE Servers: Once the TCP port line is inserted into the *tcp_services_file* on hostA and hostB, it is necessary to start an administrative listener process or daemon, db2cclst, on all hosts that participate in the instance. You can do so manually from the command line, or configure the system to automatically invoke db2cclst every time the system boots:

Manual:

From the ID of the instance you wish to administer, db2inst, invoke the following command from either hostA or hostB:

```
rah '<install_path>/bin/db2cclst'
```

For example, on AIX this command invocation would appear as:

```
rah '/usr/lpp/<v_r_ID>/bin/db2cclst'
```

The `rah` command is found in the `instance` subdirectory in the version and release subdirectory. The exact name of the version and release subdirectory varies by operating system. `instance` is the home directory of the instance you wish to use.

In this case, `<v_r_ID>` is the platform-specific version and release identifier. For example in DB2 UDB EEE for AIX in Version 5.2, the `<v_r_ID>` is `db2_05_00`.

Automatic:

As root, add an appropriate entry to the

```
/etc/inittab
```

file. For example, on AIX this command invocation should be run on hostA and hostB:

```
mkitab "db2cclst::once:su - db2inst -c
   /usr/lpp/<v_r_ID>/bin/db2cclst"
```

Every time either machine boots, db2cclst is invoked without user intervention.

In the table, `<v_r_ID>` is the platform-specific version and release identifier. For example in DB2 UDB EEE for AIX in Version 5.2, the `<v_r_ID>` is `db2_05_00`.

To verify that the listener daemon is active on each host, the following command can be invoked from the instance ID, db2inst:

```
rah 'ps -ef | grep db2cclst'
```

If you do not find the db2cclst process running on each host, additional diagnostic information can be obtained by adding the following line to `/etc/syslog.conf` on each host:

```
*.info  /tmp/db2/user.info
```

where the file `/tmp/db2/user.info` can be replaced with a more appropriate file.

Note: The file must exist and the SYSLOG daemon must be asked to re-read its configuration file after the changes are made:

```
kill -1 <syslogd PID>
```

where `syslogd PID` can be obtained by executing

```
ps -ef | grep syslogd
```

Then, after manually invoking the listener as described above, you can view the syslog file `/tmp/db2/user.info` on the failing host for error messages generated by db2cclst.

Internode Administrative Communications: Windows NT DB2 EEE Servers: The DB2 Remote Command Service (db2rcmd.exe) automatically handles internode administrative communications. In the event that a failure does occur, the Windows NT registry will contain diagnostic information.

Security: In order for the DAS to perform some administrative tasks against an instance, it must possess sufficient authority. In particular, the DAS must be a System Administrator (SYSADM) for the target, administered instance.

It is necessary to grant the DAS such authority for all DB2 instances that it will administer. Candidate instances are those which are installed on the same machine as the DAS. For a DB2 EEE instance, at least one database partition server must be present on the same machine as the DAS for it to be eligible as described above.

For example on UNIX, one way in which db2as can be granted the required authority to administer db2inst is to ensure that the primary groups of db2inst and db2as are identical. Alternatively, it is sufficient to make the primary group of db2inst a secondary group of db2as, and the primary group of db2as a secondary group of db2inst. Finally, another option would be to set the SYSADM_GROUP database administration configuration parameter for db2inst to the primary group of db2as.

On Windows NT, db2as must be a member of the Local Administrators group on hostA and hostB. In addition to the option of creating the db2as ID and adding it to the Local Administrators group on both hosts, you could create a domain ID for db2as and add this domain ID to the Local Administrators group on each host.

Environment: Installation for the DAS should configure certain registry variables that are necessary for proper operation. To verify the current values for these variables, execute the following command from either the DB2 instance ID, db2inst, or the DAS ID, db2das:

```
db2set -g
```

At least the following parameters must be defined with the following values:

```
DB2SYSTEM=hostA
DB2ADMINSERVER=db2as
```

As well, in order to communicate with the DAS from the Control Center, ensure that the DB2COMM registry variable is set to "TCPIP". To verify this setting, execute the following command from the DAS ID, db2as, and check at the global (`-g`) and instance (`-i`) levels (only one need be set):

```
db2set -all
```

Along the same lines, verify that the DB2COMM parameter is set to "TCPIP" for the DB2 instance to enable communications between the Control Center and db2inst by issuing the following command from the db2inst ID:

```
db2set -all
```

If you modify this parameter for the DAS, then you must restart the DAS for the change to take effect. Restart of the DB2 instance is also required if this parameter is modified for the DB2 instance. For db2inst, you would issue a *db2stop* followed by a *db2start*, whereas *db2admin stop* and *db2admin start* would be issued for the DAS.

Discovery of Administration Servers, Instances, and Databases: Known Discovery allows you to discover instances and databases on systems that are known to your client, and add new systems so that their instances and databases can be discovered. Search Discovery provides all of the facilities of Known Discovery and adds the option to allow your local network to be searched for other DB2 servers.

To have a server support Known Discovery, set the *discover* parameter in the DAS configuration file to `KNOWN`. To have it support Search Discovery, set this parameter to `SEARCH`. To prevent discovery of a server, and all of its instances and databases, set this parameter to `DISABLE`.

Note: The TCP/IP host name returned to a client by Search Discovery is the same host name that is returned by your DB2 server system when you enter the **hostname** command. On the client, the IP address that this host name maps to is determined by either the TCP/IP domain name server (DNS) configured on your client machine or, if no DNS is configured, a mapping entry in the client's `hosts` file. If you have multiple adapter cards configured on your DB2 server system, you must ensure that TCP/IP is configured on the server to return the correct hostname, and that the DNS or local client's `hosts` file, maps the hostname to the IP address desired.

On the client, enabling Discovery is also done using the *discover* parameter; however, in this case, the *discover* parameter is set in the client instance (or server acting as a client) as follows:

- **KNOWN**

Allows the Client Configuration Assistant to refresh systems in the known list, and to add new systems to the list by using the **Add Systems** push button. When the *discover* parameter is set to KNOWN, the Client Configuration Assistant will not be able to search the network.

- **SEARCH**

  Enables all of the facilities of Known Discovery, and enables network searching.

  The "Other Systems (Search the network)" icon only appears if this choice is made. This is the default setting.

- **DISABLE**

  Disables Discovery. In this case, the **Search the network** option is not available in the "Add Database Wizard".

Note: The *discover* parameter defaults to SEARCH on all client and server instances. The *discover* parameter defaults to SEARCH on all DB2 Administration Servers (DAS) except DAS installed in a UNIX Enterprise - Extended Edition environment, where *discover* defaults to KNOWN.

Additional Settings for Search Discovery: Search Discovery requires that the *discover_comm* parameter be set on both the server (in the DB2 Administration Server's configuration file) and the client (in the database manager configuration file).

The *discover_comm* parameter is used to control the communications protocols that the server will listen to for search requests from clients, and that clients will use to send out search requests. The *discover_comm* parameter can be set to TCP/IP or NetBIOS. Only these protocols are currently supported.

On the DAS, the values specified for *discover_comm* must be equal to, or a subset of, the values set for DB2COMM.

Note: To avoid problems with the Control Center and the Client Configuration Assistant, ensure that the DB2COMM registry variable is set in the DB2 registry using the **db2set** command. It is not recommended that you use any other method to set the DB2COMM registry variable.

On the server, the *discover_comm* parameter is set in the DAS configuration file. On the client (or a server acting as a client), *discover_comm* is set in the database manager configuration file.

Note: When using Search Discovery, at least one protocol specified by the *discover_comm* parameter on the client must match those specified by

the *discover_comm* parameter on the DAS. If there is no match, the server will not respond to the client's requests.

To check the settings for the DB2COMM registry variable, enter:

```
db2set db2comm
```

In addition, two DB2 profile registry variables can be used to tune Search Discovery via NetBIOS on the client: DB2DISCOVERYTIME and DB2NBDISCOVERYRECVBUFS. The default values for these registry variables should be suitable in most cases.

The DB2DISCOVERYTIME and DB2NBDISCOVERRCVBUFS profile registry variables are set in the client instance (or a server acting as a client). Set the registry variables as follows:

- To set the DB2DISCOVERYTIME registry value to 60 seconds, enter the following command:

  ```
  db2set db2discoverytime=60
  ```

  This specifies that Search Discovery should wait 60 seconds for a response from servers.
- To set the DB2NBDISCOVERRCVBUFS registry value to 20, enter:

  ```
  db2set db2nbdiscoverrcvbufs=20
  ```

  This specifies the number of NetBIOS buffers that will be allocated for concurrent response messages from discovered servers.

Hiding Server Instances and Databases from Discovery: You may have multiple instances, and multiple databases within these instances, on a server. You may want to hide some of these from the Discovery process.

To allow clients to discover server instances on a system, set the *discover_inst* database manager configuration parameter in each server instance on the system to `ENABLE` (this is the default value). Set this parameter to `DISABLE` to hide this instance and its databases from Discovery.

To allow a database to be discovered from a client, set the *discover_db* database configuration parameter to `ENABLE` (this is the default value). Set this parameter to `DISABLE` to hide the database from Discovery.

Setting Discovery Parameters: The *discover* and *discover_comm* parameters are set in the DAS configuration file on the server system, and in the database manager configuration file on the client. Set the parameters as follows:

- On the DAS:

  Update the DAS configuration file using the command process:

```
update admin cfg using discover [ DISABLE | KNOWN |
   SEARCH ]
update admin cfg using discover_comm [ NETBIOS | TCPIP ]
```

Stop and restart the DAS by entering the following commands:

```
db2admin stop
db2admin start
```

Note: Search Discovery will only operate on NetBIOS and TCP/IP.

- Using the Control Center:

1. Start the Client Configuration Assistant.
2. Click on the **Client Settings** push button.
3. Select the **Communications** tab.
4. Select the parameters that you want to modify from the **Parameters** window.
5. Select a value for the parameter that you want to modify from the **Value** box.
6. Click on the **OK** push button to close the **Client Settings** windows. A DB2 message window opens.
7. Click on the **OK** push button and restart your applications so that your changes can take effect.

Note: If the *discover_comm* includes NETBIOS, you must ensure that the Workstation name (*nname*) parameter is set for both the client and the DAS. Also, you must ensure that the DB2NBADAPTERS registry variable is set to the Adapter number that you want to use.

Use the Control Center to set the *discover_inst* and *discover_db* parameters:

1. Expand the object tree until you find the **Instances** folder.
2. Right-click the instance, and select **Configure** from the pop-up menu.
3. On the "Environment" page, select the *discover_inst* parameter.
4. To allow the server instance to be discovered from a client, select **Enable** and click **OK**.
5. Right-click on the database in the object tree, and select **Configure** from the pop-up menu.
6. On the "Environment" page, select the *discover_db* parameter.
7. To allow the database to be discovered from a client, select **Enable** and click **OK**.

Setting Up the DAS to Use the Client Configuration Assistant and the Control Center

You must configure DB2 Discovery to retrieve information about systems on your network. DB2 Discovery is a feature that is used by the Client Configuration Assistant and Control Center. Configuring for this feature may

require you to update instance lists and the DB2 Administration Server (DAS) configuration to ensure that DB2 Discovery retrieves the correct information.

Update Instance Lists: A DB2 Administration Server (DAS) may not be aware of all the instances in a partitioned database system because initially when an instance is created, only the DAS on the instance-owning machine is aware of the instance.

If you created an instance on a machine that does not have a DAS, you can create a DAS on this machine to make the instance known.

Perform the following steps if you created more than one DAS, and you want each DAS to be aware of all the instances in your partitioned database system:

1. **For each DAS**

   Run the **db2ilist** command on the Administration Server machine to display a list of instances known to this DAS.

   Note: If the list of instances is complete, you do not need to carry out the remaining steps but can proceed to the next section.

2. **For each instance that is missing from the instance list in the previous step**

   On the instance-owning machine, run the **db2nlist** command to see if there is an entry for the machine that has the DAS. If there is not, you must run the **db2ncrt** command to add this machine to the instance.

   Note: The network shared drive for the instance must be available on the DAS machine.

Update the DAS Configuration

By default, the setup program sets the DB2SYSTEM registry variable to the Windows NT computer name. The system names that are retrieved by Discovery are the systems on which a DB2 Administration Server (DAS) resides. Discovery uses these systems as coordinator nodes when connections are established.

There are two ways of updating a DAS configuration:

- If you want to be able to select a coordinator node from a list of DB2 systems, set DISCOVER=SEARCH (which is the default) in each DB2 Administration Server's configuration file.

  When there are multiple DAS present, the same instance may appear in more than one system on the Client Configuration Assistant or Control Center's interface; however, each system will have a different communications access path to instances. Users can select different DB2 systems as coordinator nodes for communications and thereby redistribute the workload.

- If you do not want users to be able to select the coordinator node, set DISCOVER=KNOWN on all DAS, except set DISCOVER=SEARCH on just one DAS in the DAS configuration. Discovery uses the database partition server where the DAS resides as a coordinator node when connections are established.

Creating a Node Configuration File

If your database is to operate in a partitioned database environment, you must create a node configuration file called db2nodes.cfg. This file must be located in the sqllib subdirectory of the home directory for the instance before you can start the database manager with parallel capabilities across multiple partitions. The file contains configuration information for all database partitions in an instance, and is shared by all database partitions for that instance.

Windows NT Considerations: If you are using DB2 Enterprise - Extended Edition on Windows NT, the node configuration file is created for you when you create the instance. You should not attempt to create or modify the node configuration file manually.

Note: You should not create files or directories under the sqllib subdirectory other than those created by DB2 to prevent the loss of data if an instance is deleted. There are two exceptions. If your system supports stored procedures, put the stored procedure applications in the function subdirectory under the sqllib subdirectory. (For information on stored procedures, refer to "Stored Procedures" in *Administration Guide: Performance*.) The other exception is when user-defined distinct functions (UDFs) have been created. UDF executables are allowed in the same directory.

The file contains one line for each database partition that belongs to an instance. Each line has the following format:

```
nodenum hostname [logical-port [netname]]
```

Tokens are delimited by blanks. The variables are:

nodenum

The node number, which can be from 0 to 999, uniquely defines a node. Node numbers must be in ascending sequence. You can have gaps in the sequence.

Once a node number is assigned, it cannot be changed. (Otherwise the information in the partitioning map, which specifies how data is partitioned, would be compromised.)

If you drop a node, its node number can be used again for any new node that you add.

The node number is used to generate a node name in the database directory. It has the format:

NODE*nnnn*

The *nnnn* is the node number, which is left-padded with zeros. This node number is also used by the CREATE DATABASE and DROP DATABASE commands.

hostname
The hostname of the IP address for inter-partition communications. (There is an exception when `netname` is specified. In this situation, `netname` is used for most communications, with `hostname` only being used for DB2START, DB2STOP, and `db2_all`.)

logical-port
This parameter is optional, and specifies the logical port number for the node. This number is used with the database manager instance name to identify a TCP/IP service name entry in the `etc/services` file.

The combination of the IP address and the logical port is used as a well-known address, and must be unique among all applications to support communications connections between nodes.

For each *hostname*, one *logical-port* must be either 0 (zero) or blank (which defaults to 0). The node associated with this *logical-port* is the default node on the host to which clients connect. You can override this with the DB2NODE environment variable in `db2profile` script, or with the sqlesetc() API.

If you have multiple nodes on the same host (that is, more than one *nodenum* for a host), you should assign the *logical-port* numbers to the logical nodes in ascending order, from 0, with no gaps.

netname
This parameter is optional, and is used to support a host that has more than one active TCP/IP interface, each with its own hostname.

The following example shows a possible node configuration file for an RS/6000 SP system on which SP2EN1 has multiple TCP/IP interfaces, two logical nodes, and uses SP2SW1 as the DB2 Universal Database interface. It also shows the node numbers starting at 1 (rather than at 0), and a gap in the *nodenum* sequence:

```
nodenum   hostname   logical-port   netname
1         SP2EN1     0              SP2SW1
2         SP2EN1     1              SP2SW1
4         SP2EN2     0
5         SP2EN3
```

You can update the db2nodes.cfg file using an editor of your choice. (The exception is: an editor should not be used on Windows NT.) You must be careful, however, to protect the integrity of the information in the file, as data partitioning requires that the node number not be changed. The node configuration file is locked when you issue DB2START and unlocked after DB2STOP ends the database manager. The DB2START command can update the file, if necessary, when the file is locked. For example, you can issue DB2START with the RESTART option or the ADDNODE option.

Note: If the DB2STOP command is not successful and does not unlock the node configuration file, issue DB2STOP FORCE to unlock it.

Creating the Database Configuration File

A *database configuration file* is also created for each database. The creation of this file is done for you. This file contains values for various *configuration parameters* that affect the use of the database, such as:

- Parameters specified and/or used when creating the database (for example, database code page, collating sequence, DB2 release level)
- Parameters indicating the current state of the database (for example, backup pending flag, database consistency flag, roll-forward pending flag)
- Parameters defining the amount of system resources that the operation of the database may use (for example, buffer pool size, database logging, sort memory size).

These parameters are described in detail in "Configuring DB2" found in *Administration Guide: Performance*.

You should not manually change the parameters in the configuration file. You should only use the supported interface.

Performance Tip: Many of the configuration parameters come with default values, but may need to be updated to achieve optimal performance for your database.

For multiple partitions: When you have a database that is partitioned across more than one partition, the configuration file should be the same on all database partitions. Consistency is required since the SQL compiler compiles distributed SQL statements based on information in the local node configuration file and creates an access plan to satisfy the needs of the SQL statement. Maintaining different configuration files on database partitions

could lead to different access plans, depending on which database partition the statement is prepared. Use **db2_all** to keep the configuration files synchronized across all database partitions.

Replicating Configuration Information Using Response Files

A response-file generator utility called *db2rspgn* is available to create a response file that can be used when re-installing your system or when you wish to replicate to identical system the registry variables, database manager configuration parameters, and database administration configuration parameters of your current system.

After having installed a system with one or more DB2 products, and after tuning parameters for the environment, you can use *db2rspgn* to generate the required values into a response file. The response file can then be used to re-create the identical system.

The command line syntax declares the destination directory for the response file(s) and any supporting files. In addition, you can optionally specify the instances you wish copied; and, you can optionally disable the administration instance and/or the DataLinks server instance. See *Administering Satellites Guide and Reference* for more information on mass deployment issues.

Refer to the appropriate *Quick Beginnings* to see the details on the syntax of this utility and a discussion on how to use the generated response files.

Enabling FCM Communications

In a partitioned database environment, most communication between database partitions is handled by the Fast Communications Manager (FCM). To enable the FCM at a database partition and allow communication with other database partitions, you must create a service entry in the partition's `services` file of the `etc` directory as shown below. The FCM uses the specified port to communicate. If you have defined multiple partitions on the same host, you must define a range of ports as shown below.

Windows NT Considerations

If you are using DB2 Enterprise - Extended Edition in the Windows NT environment, the TCP/IP port range is automatically added to the services file by:

- The install program when it creates the instance or adds a new node
- The db2icrt utility when it creates a new instance
- The db2ncrt utility when it adds the first node on the machine

For additional information, refer to the *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

The syntax of a service entry is as follows:

```
DB2_instance port/tcp #comment
```

DB2_*instance*
The value for *instance* is the name of the database manager instance. All characters in the name must be lowercase. Assuming an instance name of `db2puser`, you would specify `DB2_db2puser`

port/**tcp**
The TCP/IP port that you want to reserve for the database partition.

#*comment*
Any comment that you want to associate with the entry. The comment must be preceded by a pound sign (#).

If the `/etc/services` file is shared, you must ensure that the number of ports allocated in the file is either greater than or equal to the largest number of multiple database partitions in the instance. When allocating ports, also ensure that you account for any processor that can be used as a backup.

If the `/etc/services` file is not shared, the same considerations apply, with one additional consideration: you must ensure that the entries defined for the DB2 instance are the same in all `/etc/services` files (though other entries that do not apply to your partitioned database do not have to be the same).

If you have multiple database partitions on the same host in an instance, you must define more than one port for the FCM to use. To do this, include two lines in the `etc/services` file to indicate the range of ports you are allocating. The first line specifies the first port, while the second line indicates the end of the block of ports. In the following example, five ports are allocated for the instance `sales`. This means no processor in the instance has more than five database partitions.

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

Note: You must specify `END` in uppercase only. Also you must ensure that you include both underscore (_) characters.

Chapter 3. Creating a Database

This chapter provides a brief look at each of the various objects that may be part of the implementation of your database design.

The previous chapter focused on the information you need to know before creating a database. That chapter also covered several topics and tasks you must perform before creating a database.

The final chapter in this part presents what you must consider before altering a database. In addition, the chapter explains how to alter or drop database objects.

When you create a database, each of the following tasks are done for you:

- Setting up of all the system catalog tables that are needed by the database
- Allocation of the database recovery log
- Creation of the database configuration file and the default values are set
- Binding of the database utilities to the database

The following database privileges are automatically granted to PUBLIC: CREATETAB, BINDADD, CONNECT, IMPLICIT_SCHEMA, and SELECT privilege on the system catalog views.

To create a database using the Control Center:

1. Expand the object tree until you find the **Databases** folder.
2. Right-click the **Databases** folder, and select **Create —> Database Using Wizard** from the pop-up menu.
3. Follow the steps to complete this task.

The following command line processor command creates a database called `personl`, in the default location, with the associated comment "Personnel DB for BSchiefer Co".

```
create database personl
   with "Personnel DB for BSchiefer Co"
```

If you want to create a database in a different, possibly remote, database manager instance, see "Using Multiple Instances of the Database Manager" on page 53. This topic also provides an introduction to the command you need to use if you want to perform any instance-level administration against an instance other than your default instance, including remote instances.

Note: Refer to the *Command Reference* for information about the default database location and about specifying a different location with the CREATE DATABASE command.

The tasks carried out by you, or done for you by the database manager, when you create a database are discussed in the following sections:

- "Definition of Initial Nodegroups" on page 103
- "Definition of Initial Table Spaces" on page 103
- "Definition of System Catalog Tables" on page 104
- "Definition of Database Directories" on page 105
- "DCE Directory Services" on page 107
- "Lightweight Directory Access Protocol (LDAP) Directory Services" on page 107
- "Definition of Database Recovery Log" on page 109
- "Binding Utilities to the Database" on page 109
- "Cataloging a Database" on page 109
- "Creating Nodegroups" on page 108
- "Creating a Table Space" on page 111
- "Creating a Schema" on page 116
- "Creating and Populating a Table" on page 118
- "Creating a Trigger" on page 136
- "Creating a User-Defined Function (UDF) or Method" on page 138
- "Creating a User-Defined Type (UDT)" on page 142
- "Creating a View" on page 144
- "Creating a Summary Table" on page 147
- "Creating an Alias" on page 149
- "Creating a Wrapper" on page 151
- "Creating a Server" on page 152
- "Creating a Nickname" on page 159
- "Creating an Index, Index Extension, or an Index Specification" on page 161

For additional information related to the physical implementation of your database, refer to *Administration Guide: Planning*.

Definition of Initial Nodegroups

When a database is initially created, database partitions are created for all partitions specified in the db2nodes.cfg file. Other partitions can be added or removed with the ADD NODE and DROP NODE commands.

Three nodegroups are defined:

- IBMCATGROUP for the SYSCATSPACE table space, holding system catalog tables
- IBMTEMPGROUP for the TEMPSPACE1 table space, holding temporary tables created during database processing
- IBMDEFAULTGROUP for the USERSPACE1 table space, by default holding user tables and indexes.

Definition of Initial Table Spaces

When a database is created, three table spaces are defined:

- SYSCATSPACE for the system catalog tables (see "Definition of System Catalog Tables" on page 104)
- TEMPSPACE1 for system temporary tables created during database processing
- USERSPACE1 for user-defined tables and indexes

Note: When you first create a database no user temporary table space is created.

If you do not specify any table space parameters with the CREATE DATABASE command, the database manager creates these table spaces using system managed storage (SMS) directory containers. These directory containers are created in the subdirectory created for the database (refer to *Administration Guide: Planning* for more information on database physical directories). The extent size for these table spaces is set to the default.

To define initial table spaces using the Control Center:

1. Expand the object tree until you see the **Databases** folder.
2. Right-click the **Databases** folder, and select **Create —> Database Using Wizard** from the pop-up menu.
3. Follow the steps to complete this task.

To define initial table spaces using the command line, enter:

```
CREATE DATABASE <name>
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('<path>')
```

```
  EXTENTSIZE <value> PREFETCHSIZE <value>
USER TABLESPACE
  MANAGED BY DATABASE USING (FILE'<path>' 5000,
                             FILE'<path>' 5000)
  EXTENTSIZE <value> PREFETCHSIZE <value>
TEMPORARY TABLESPACE
  MANAGED BY SYSTEM USING ('<path>')
WITH "<comment>"
```

If you do not want to use the default definition for these table spaces, you may specify their characteristics on the CREATE DATABASE command. For example, the following command could be used to create your database on OS/2:

```
CREATE DATABASE PERSONL
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
    EXTENTSIZE 16 PREFETCHSIZE 32
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                               FILE'd:\db2data\personl' 5000)
    EXTENTSIZE 32 PREFETCHSIZE 64
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('f:\db2temp\personl')
  WITH "Personnel DB for BSchiefer Co"
```

In this example, the definition for each of the initial table spaces is explicitly provided. You only need to specify the table space definitions for those table spaces for which you do not want to use the default definition.

The coding of the MANAGED BY phrase on the CREATE DATABASE command follows the same format as the MANAGED BY phrase on the CREATE TABLESPACE command. For additional examples, see "Creating a Table Space" on page 111.

Refer to the *Administration Guide: Planning* manual and the information on designing and choosing table spaces before creating your database.

Definition of System Catalog Tables

A set of system catalog tables is created and maintained for each database. These tables contain information about the definitions of the database objects (for example, tables, views, indexes, and packages), and security information about the type of access that users have to these objects. These tables are stored in the SYSCATSPACE table space.

These tables are updated during the operation of a database; for example, when a table is created. You cannot explicitly create or drop these tables, but

you can query and view their content. When the database is created, in addition to the system catalog table objects, the following database objects are defined in the system catalog:

- A set of user-defined functions (UDFs) is created in the SYSFUN schema. For more information about these system-created functions, refer to the *SQL Reference*.
- A set of read-only views for the system catalog tables is created in the SYSCAT schema. Refer to "Catalog Views" in the *SQL Reference* for information about these views.
- A set of updatable catalog views is created in the SYSSTAT schema. These updatable views allow you to update certain statistical information to investigate the performance of a hypothetical database, or to update statistics without using the RUNSTATS utility. Refer to "Updatable Catalog Views" in the *SQL Reference* for information about these views.

After your database has been created, you may wish to limit the access to the system catalog views, as described in "Securing the System Catalog Views" on page 270.

Definition of Database Directories

Three directories are used when establishing or setting up a new database.

- Local Database Directory
- System Database Directory
- Node Directory

Local Database Directory

A *local database directory* file exists in each path (called a "drive" on some operating systems) in which a database has been defined. This directory contains one entry for each database accessible from that location. Each entry contains:

- The database name provided with the CREATE DATABASE command
- The database alias name (which is the same as the database name, if an alias name is not specified)
- A comment describing the database, as provided with the CREATE DATABASE command
- The name of the root directory for the database
- Other system information.

To see the contents of this file for a particular database, issue the following command, where *location* specifies the location of the database:

```
LIST DATABASE DIRECTORY ON location
```

System Database Directory

A *system database directory* file exists for each instance of the database manager, and contains one entry for each database that has been cataloged for this instance. Databases are implicitly cataloged when the CREATE DATABASE command is issued and can also be explicitly cataloged with the CATALOG DATABASE command. For information about cataloging databases, see "Cataloging a Database" on page 109.

For each database created, an entry is added to the directory containing the following information:

- The database name provided with the CREATE DATABASE command
- The database alias name (which is the same as the database name, if an alias name is not specified)
- The database comment provided with the CREATE DATABASE command
- The location of the *local database directory*
- An indicator that the database is *indirect*, which means that it resides on the same machine as the system database directory file
- Other system information.

To see the contents of this file, issue the LIST DATABASE DIRECTORY command *without* specifying the location of the database directory file.

In a partitioned database environment, you must ensure that all database partitions always access the same system database directory file, `sqldbdir`, in the `sqldbdir` subdirectory of the home directory for the instance. Unpredictable errors can occur if either the system database directory or the system intention file `sqldbins` in the same `sqldbdir` subdirectory are symbolic links to another file that is on a shared file system. These files are described in "Enabling Data Partitioning" on page 57.

Node Directory

The database manager creates the node directory when the first database partition is cataloged. To catalog a database partition, use the CATALOG NODE command. To list the contents of the local node directory, use the LIST NODE DIRECTORY command. The node directory is created and maintained on each database client. The directory contains an entry for each remote workstation having one or more databases that the client can access. The DB2 client uses the communication end point information in the node directory whenever a database connection or instance attachment is requested.

The entries in the directory also contain information on the type of communication protocol to be used to communicate from the client to the remote database partition. Cataloging a local database partition creates an

alias for an instance that resides on the same machine. A local node should be cataloged when there is more than one instance on the same workstation to be accessed from the user's client.

DCE Directory Services

DCE is an Open Systems Foundation** (OSF**) architecture that provides tools and services to support the creation, use, and maintenance of applications in a distributed heterogeneous computing environment. It is a layer between the operating system, the network, and a distributed application that allows client applications to access remote servers.

With local directories, the physical location of the target database is individually stored on each client workstation in the database directory and node directory. The database administrator can therefore spend a large amount of time updating and changing these directories. The DCE directory services provide a central directory alternative to the local directories. It allows information about a database or a database manager instance to be recorded once in a central location, and any changes or updates to be made at that one location.

DCE is not a prerequisite for running DB2, but if you are operating in a DCE environment, see "Appendix B. Using Distributed Computing Environment (DCE) Directory Services" on page 319 for more information.

Lightweight Directory Access Protocol (LDAP) Directory Services

Lightweight Directory Access Protocol (LDAP) is an industry standard access method to directory services. A directory service is a repository of resource information about multiple systems and services within a distributed environment; and it provides client and server access to these resources. Each database server instance will publish its existence to an LDAP server and provide database information to the LDAP directory when the databases are created. When a client connects to a database, the catalog information for the server can be retrieved from the LDAP directory. Each client is no longer required to store catalog information locally on each machine. Client applications search the LDAP directory for information required to connect to the database.

LDAP is not a prerequisite for running DB2, but if you are operating in an LDAP environment, see "Appendix J. Lightweight Directory Access Protocol (LDAP) Directory Services" on page 395 for more information.

Creating Nodegroups

You create a nodegroup with the CREATE NODEGROUP statement. This statement specifies the set of nodes on which the table space containers and table data are to reside. This statement also:

- Creates a partitioning map for the nodegroup. For details about the partitioning map, refer to *Administration Guide: Planning*.
- Generates a partitioning map ID.
- Inserts records into the following catalog tables:
  - SYSCAT.NODEGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.NODEGROUPDEF

To create a nodegroup using the Control Center:

1. Expand the object tree until you see the **Nodegroups** folder.
2. Right-click the **Nodegroups** folder, and select **Create** from the pop-up menu.
3. On the Create Nodegroups window, complete the information, use the arrows to move nodes from the **Available nodes** box to the **Selected nodes** box, and click **Ok**.

To create a nodegroup using the command line, enter:

```
CREATE NODEGROUP <name> ON NODES (<value>,<value>)
```

Assume that you want to load some tables on a subset of the database partitions in your database. You would use the following command to create a nodegroup of two nodes (1 and 2) in a database consisting of at least three (0 to 2) nodes:

```
CREATE NODEGROUP mixng12 ON NODES (1,2)
```

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

The CREATE DATABASE command or sqlecrea() API also create the default system nodegroups, IBMDEFAULTGROUP, IBMCATGROUP, and IBMTEMPGROUP. (Refer to *Administration Guide: Planning* for more information on nodegroups.)

Definition of Database Recovery Log

A *database recovery log* keeps a record of all changes made to a database, including the addition of new tables or updates to existing ones. This log is made up of a number of *log extents*, each contained in a separate file called a *log file*.

The database recovery log can be used to ensure that a failure (for example, a system power outage or application error) does not leave the database in an inconsistent state. In case of a failure, the changes already made but not committed are rolled back, and all committed transactions, which may not have been physically written to disk, are redone. These actions ensure the integrity of the database.

For more information, see *Data Movement Utilities Guide and Reference*.

Binding Utilities to the Database

When a database is created, the database manager attempts to bind the utilities in `db2ubind.lst` to the database. This file is stored in the `bnd` subdirectory of your `sqllib` directory.

Binding a utility creates a *package*, which is an object that includes all the information needed to process specific SQL statements from a single source file.

Note: If you wish to use these utilities from a client, you must bind them explicitly. Refer to the *Quick Beginnings* manual appropriate to your platform for information.

If for some reason you need to bind or rebind the utilities to a database, issue the following commands using the command line processor:

```
connect to sample
bind @db2ubind.lst
```

Note: You must be in the directory where these files reside to create the packages in the `sample` database. The bind files are found in the `BND` subdirectory of the `SQLLIB` directory. In this example, `sample` is the name of the database.

Cataloging a Database

When you create a new database, it is automatically cataloged in the system database directory file. You may also use the CATALOG DATABASE command to explicitly catalog a database in the system database directory file. The CATALOG DATABASE command allows you to catalog a database with a

different alias name, or to catalog a database entry that was previously deleted using the UNCATALOG DATABASE command.

The following command line processor command catalogs the `personl` database as `humanres`:

```
catalog database personl as humanres
   with "Human Resources Database"
```

Here, the system database directory entry will have `humanres` as the database alias, which is different from the database name (`personl`).

You can also catalog a database on an instance other than the default. In the following example, connections to database `B` are to `INSTANCE_C`.

```
catalog database b as b at node instance_c
```

Note: The CATALOG DATABASE command is also used on client nodes to catalog databases that reside on database server machines. For more information, refer to the *Quick Beginnings* manual appropriate to your platform.

For information on the Distributed Computing Environment (DCE) cell directory, see "DCE Directory Services" on page 107 and "Appendix B. Using Distributed Computing Environment (DCE) Directory Services" on page 319.

Note: To improve performance, you may cache directory files, including the database directory, in memory. (Refer to "Directory Cache Support (dir_cache)" in the *Administration Guide: Performance* for information about enabling directory caching.) When directory caching is enabled, a change made to a directory (for example, using a CATALOG DATABASE or UNCATALOG DATABASE command) by another application may not become effective until your application is restarted. To refresh the directory cache used by a command line processor session, issue a `db2 terminate` command.

In addition to the application level cache, a database manager level cache is also used for internal, database manager look-up. To refresh this "shared" cache, issue the `db2stop` and `db2start` commands.

For more information about directory caching, refer to "Directory Cache Support (dir_cache)" in the *Administration Guide: Performance*.

Creating a Table Space

Creating a table space within a database assigns containers to the table space and records its definitions and attributes in the database system catalog. You can then create tables within this table space.

Refer to *Administration Guide: Planning* for design information on table spaces.

The syntax of the CREATE TABLESPACE statement is discussed in detail in the *SQL Reference*. For information on SMS and DMS table spaces, refer to the *Administration Guide: Planning*.

To create a table space using the Control Center:

1. Expand the object tree until you see the **Table spaces** folder.
2. Right-click the **Table spaces** folder, and select **Create —> Table Space Using Wizard** from the pop-up menu.
3. Follow the steps in the wizard to complete your task.

To create an SMS table space using the command line, enter:

```
CREATE TABLESPACE <NAME>
   MANAGED BY SYSTEM
   USING ('<path>')
```

To create a DMS table space using the command line, enter:

```
CREATE TABLESPACE <NAME>
   MANAGED BY DATABASE
   USING (FILE'<path>' <size>)
```

The following SQL statement creates an SMS table space on OS/2 or Windows NT using three directories on three separate drives:

```
CREATE TABLESPACE RESOURCE
   MANAGED BY SYSTEM
   USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
```

The following SQL statement creates a DMS table space on OS/2 using two file containers each with 5,000 pages:

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (FILE'd:\db2data\acc_tbsp' 5000,
          FILE'e:\db2data\acc_tbsp' 5000)
```

In the above two examples, explicit names have been provided for the containers. However, if you specify relative container names, the container is created in the subdirectory created for the database (refer to *Administration Guide: Planning* for more information on database physical directories).

In addition, if part of the path name specified does not exist, the database manager creates it. If a subdirectory is created by the database manager, it may also be deleted by the database manager when the table space is dropped.

The assumption in the above examples is that the table spaces are not associated with a specific nodegroup. The default nodegroup IBMDEFAULTGROUP is used when the following parameter is not specified in the statement:

```
IN nodegroup
```

The following SQL statement creates a DMS table space on a UNIX-based system using three logical volumes of 10 000 pages each, and specifies their I/O characteristics:

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (DEVICE '/dev/rdblv6' 10000,
          DEVICE '/dev/rdblv7' 10000,
          DEVICE '/dev/rdblv8' 10000)
   OVERHEAD 24.1
   TRANSFERRATE 0.9
```

The UNIX devices mentioned in this SQL statement must already exist, and the instance owner and the SYSADM group must be able to write to them.

The following example creates a DMS table space on a nodegroup called ODDNODEGROUP in a UNIX partitioned database. ODDNODEGROUP must be previously created with a CREATE NODEGROUP statement. In this case, the ODDNODEGROUP nodegroup is assumed to be made up of database partitions numbered 1, 3, and 5. On all database partitions, use the device `/dev/hdisk0` for 10 000 4 KB pages. In addition, declare a device for each database partition of 40 000 4 KB pages.

```
CREATE TABLESPACE PLANS
   MANAGED BY DATABASE
   USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000) ON NODE 1
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000) ON NODE 3
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000) ON NODE 5
```

UNIX devices are classified into two categories: character serial devices and block-structured devices. For all file-system devices, it is normal to have a corresponding character serial device (or *raw* device) for each block device (or *cooked* device). The block-structured devices are typically designated by names similar to “hd0” or “fd0”. The character serial devices are typically designated by names similar to “rhd0”, “rfd0”, or “rmt0”. These character serial devices have faster access than block devices. The character serial device names should be used on the CREATE TABLESPACE command and not block device names.

The overhead and transfer rate help to determine the best access path to use when the SQL statement is compiled. For information on the OVERHEAD and TRANSFERRATE parameters, refer to "Tuning Application Performance" in the *Administration Guide: Performance*.

DB2 can greatly improve the performance of sequential I/O using the sequential prefetch facility, which uses parallel I/O. Refer to "Understanding Sequential Prefetching" in the *Administration Guide: Performance* for details on this facility.

You can also create a table space that uses a page size larger than the default 4 KB size. The following SQL statement creates an SMS table space on a UNIX-based system with an 8 KB page size.

```
CREATE TABLESPACE SMS8K
   PAGESIZE 8192
   MANAGED BY SYSTEM
   USING ('FSMS_8K_1')
   BUFFERPOOL BUFFPOOL8K
```

Notice that the associated buffer pool must also have the same 8 KB page size.

The created table space cannot be used until the buffer pool it references is activated.

The ALTER TABLESPACE SQL statement can be used to add a container to a DMS table space and modify the PREFETCHSIZE, OVERHEAD, and TRANSFERRATE settings for a table space. The transaction issuing the table space statement should be committed as soon as possible, to prevent system catalog contention.

Note: The PREFETCHSIZE should be a multiple of the EXTENTSIZE. For example if the EXTENTSIZE is 10, the PREFETCHSIZE should be 20 or 30. For more information, refer to "Understanding Sequential Prefetching" in the *Administration Guide: Performance*.

Creating a System Temporary Table Space

A system temporary table space is used to store system temporary tables. When a database is created, one of the three default table spaces defined is a system temporary table space called "TEMPSPACE1".

Note: A database must always have at least one system temporary table space since system temporary tables can only be stored in such a table space.

You can use the CREATE TABLESPACE statement to create another system temporary table space. For example,

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
   MANAGED BY SYSTEM
   USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

The only nodegroup that can be specified when creating a system temporary table space is IBMTEMPGROUP.

Creating a User Temporary Table Space

A user temporary table space is used to store declared temporary tables.

You can use the CREATE TABLESPACE statement to create a user temporary table space:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
   MANAGED BY DATABASE
   USING (FILE 'd:\db2data\user_tbsp' 5000,
          FILE 'e:\db2data\user_tbsp' 5000)
```

Like regular table spaces, user temporary table spaces may be created in any nodegroup other than IBMTEMPGROUP. The default nodegroup used when creating a user temporary table space is IBMDEFAULTGROUP.

The DECLARE GLOBAL TEMPORARY TABLE statement defines declared temporary tables for use within a user temporary table space.

Creating Table Spaces in Nodegroups

By placing a table space in a multiple database partition nodegroup, all of the tables within the table space are divided or partitioned across each database partition in the nodegroup. The table space is created into a nodegroup. Once in a nodegroup, the table space must remain there; it cannot be changed to another nodegroup. The CREATE TABLESPACE statement is used to associate a table space with a nodegroup.

Raw I/O

DB2 Universal Database supports direct disk access (raw I/O). This allows you to attach a direct disk access (raw) device to any DB2 Universal Database system. (The only exceptions are the Windows 95, and Windows 98 operating systems.) The following list demonstrates the physical and logical methods for identifying this type of device:

- On Windows, to specify a physical hard drive, use the following syntax:

  ```
  \\.\PhysicalDriveN
  ```

  where N represents one of the physical drives in the system. In this case, N could be replaced by 0, 1, 2, or any other positive integer:

  ```
  \\.\PhysicalDisk5
  ```

- On Windows, to specify a logical raw partition (that is, an unformatted partition) use the following syntax:

  \\.\N:

  where N: represents a logical drive letter in the system. For example, N: could be replaced by E: or any other drive letter.
- **Note:** You must have Windows NT Version 4.0 with Service Pack 3 installed to be able to write logs to a device.
- On UNIX-based platforms, use the character serial device name; for example, `/dev/rhd0`

Using Raw I/O on Linux

Linux has a pool of raw device nodes that must be bound to a block device before raw I/O can be performed on it. There is a raw device controller that acts as the central repository of raw to block device binding information. Binding is performed using a utility named `raw`, which is normally supplied by the Linux distributor.

Before you set up raw I/O on Linux, you require the following:

- One or more free IDE or SCSI disk partitions
- Linux kernel 2.4.0 or later (However, some Linux distributions offer raw I/O on 2.2 kernels.)
- A raw device controller named `/dev/rawctl` or `/dev/raw`. If not, create a symbolic link:

  ```
  # ln -s /dev/your_raw_dev_ctrl /dev/rawctl
  ```
- The raw utility, which is usually provided with the Linux distribution
- DB2 Version 7.1 FixPak 3 or later

Note: Of the distributions currently supporting raw I/O, the naming of raw device nodes is different:

```
Distribution  Raw device nodes      Raw device controller
------------  --------------------  ---------------------
RedHat 6.2    /dev/raw/raw1 to 255  /dev/rawctl
SuSE 7.0      /dev/raw1 to 63       /dev/raw
```

DB2 supports either of the above raw device controllers, and most other names for raw device nodes. Raw devices are not supported by DB2 on Linux/390.

To configure raw I/O on Linux:

In this example, the raw partition to be used is `/dev/sda5`. It should not contain any valuable data.

Step 1. Calculate the number of 4 096-byte pages in this partition, rounding down if necessary. For example:

```
# fdisk /dev/sda
Command (m for help): p

Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot   Start    End    Blocks   Id  System
/dev/sda1            1    523   4200997   83  Linux
/dev/sda2          524   1106   4682947+   5  Extended
/dev/sda5          524   1106   4682947   83  Linux

Command (m for help): q
#
```

The number of pages in /dev/sda5 is

```
num_pages = floor( ((1106-524+1)*16065*512)/4096 )
num_pages = 11170736
```

Step 2. Bind an unused raw device node to this partition. This needs to be done every time the machine is rebooted, and requires root access. Use `raw -a` to see which raw device nodes are already in use:

```
# raw /dev/raw/raw1 /dev/sda5
/dev/raw/raw1: bound to major 8, minor 5
```

Step 3. Set global read permissions on the raw device controller and the disk partition. Set global read and write permissions on the raw device:

```
# chmod a+r /dev/rawctl
# chmod a+r /dev/sdb1
# chmod a+rw /dev/raw/raw1
```

Step 4. Create the table space in DB2, specifying the raw device, not the disk partition. For example:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 11170736)
```

Table spaces on raw devices are also supported for all other page sizes supported by DB2.

Creating a Schema

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 through the use of the CREATE SCHEMA statement. 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.

The syntax of the CREATE SCHEMA statement is described in detail in the *SQL Reference*. The new schema name cannot already exist in the system catalogs and it cannot begin with ″SYS″.

If a user has SYSADM or DBADM authority, then the user can create a schema with any valid name. When a database is created, IMPLICIT_SCHEMA authority is granted to PUBLIC (that is, to all users).

The definer of any objects created as part of the CREATE SCHEMA statement is the schema owner. This owner can GRANT and REVOKE schema privileges to other users.

This statement must be issued by a user with DBADM authority.

Schemas may also be implicitly created when a user has IMPLICIT_SCHEMA authority. With this authority, users implicitly create a schema whenever they create an object with a schema name that does not already exist.

If users do not have IMPLICIT_SCHEMA authority, the only schema they can create is one that has the same name as their own authorization ID.

Direct access to objects within a schema is not allowed since the schema is used to enforce uniqueness in the database. This becomes clear when considering the possibility that two users could create two tables (or other objects) with the same name. Without a schema to enforce uniqueness, ambiguity would exist if a third user attempted to query the table. It is not possible to determine which table to use without some further qualification.

To allow another user to access a table without entering a schema name as part of the qualification on the table name requires that a view be established for that user. The definition of the view would define the fully-qualified table name including the user's schema; the user would simply need to query using the view name. The view would be fully-qualified by the user's schema as part of the view definition.

To create a schema using the Control Center:

1. Expand the object tree until you see the **Schema** folder.
2. Right-click the **Schema** folder, and select **Create** from the pop-up menu.
3. Complete the information for the new schema, and click **Ok**.

To create a schema using the command line, enter:

```
CREATE SCHEMA <name> AUTHORIZATION <name>
```

The following is an example of a CREATE SCHEMA statement that creates a schema for an individual user with the authorization ID ″joe″:

```
CREATE SCHEMA joeschma AUTHORIZATION joe
```

Setting a Schema

You may want to establish a default schema for use by unqualified object references in dynamic SQL statements issued from within a specific DB2 connection. This is done by setting the special register CURRENT SCHEMA to the schema you wish to use as the default. Any user can set this special register: no authorization is required.

The syntax of the SET SCHEMA statement is described in detail in the *SQL Reference* manual.

The following is an example of how to set the CURRENT SCHEMA special register:

```
SET CURRENT SCHEMA = 'SCHEMA01'
```

This statement can be used from within an application program or issued interactively. Once set, the value of the CURRENT SCHEMA special register is used as the qualifier (schema) for unqualified object references in dynamic SQL statements, with the exception of the CREATE SCHEMA statement where an unqualified reference to a database object exists.

The initial value of the CURRENT SCHEMA special register is equal to the authorization ID of the current session user.

Creating and Populating a Table

After you determine how to organize your data into tables, the next step is to create those tables, by using the CREATE TABLE statement. The table descriptions are stored in the system catalog of the database to which you are connected.

The syntax of the CREATE TABLE statement is described in detail in the *SQL Reference*. For information on creating a summary table, see “Creating a Summary Table” on page 147. For information about naming tables, columns, and other database objects, see “Appendix A. Naming Rules” on page 313.

The CREATE TABLE statement gives the table a name, which is a qualified or unqualified identifier, and a definition for each of its columns. You can store each table in a separate table space, so that a table space contains only one table. If a table will be dropped and created often, it is more efficient to store it in a separate table space and then drop the table space instead of the table. You can also store many tables within a single table space. In a partitioned

database environment, the table space chosen also defines the nodegroup and the database partitions on which table data is stored.

The table does not contain any data at first. To add rows of data to it, use one of the following:

- The INSERT statement, described in the *SQL Reference*
- The LOAD or IMPORT commands, described in the *Command Reference*
- The autoloader utility if working in a partitioned database environment as described in the *Data Movement Utilities Guide and Reference*.

Details concerning the movement of data into and out of tables is presented in *Data Movement Utilities Guide and Reference*.

Adding data to a table can be done without logging the change. The NOT LOGGED INITIALLY clause on the CREATE TABLE statement prevents logging the change to the table. Any changes made to the table by an INSERT, DELETE, UPDATE, CREATE INDEX, DROP INDEX, or ALTER TABLE operation in the same unit of work in which the table is created are not logged. Logging begins in subsequent units of work.

A table consists of one or more column definitions. A maximum of 500 columns can be defined for a table. Columns represent the attributes of an entity. The values in any column are all the same type of information. Refer to the *SQL Reference* for more information.

Note: The maximum of 500 columns is true when using a 4 KB page size. The maximum is 1012 columns when using an 8 KB, 16 KB, or 32 KB page size.

A column definition includes a *column name, data type*, and any necessary *null attribute*, or default value (optionally chosen by the user).

The column name describes the information contained in the column and should be something that will be easily recognizable. It must be unique within the table; however, the same name can be used in other tables. See "Appendix A. Naming Rules" on page 313 for information about naming rules.

The data type of a column indicates the length of the values in it and the kind of data that is valid for it. The database manager uses character string, numeric, date, time and large object data types. Graphic string data types are only available for database environments using multi-byte character sets. In addition, columns can be defined with user-defined distinct types, which are discussed in "Creating a User-Defined Type (UDT)" on page 142.

The default attribute specification indicates what value is to be used if no value is provided. The default value can be specified, or a system-defined default value used. Default values may be specified for columns with, and without, the null attribute specification.

The null attribute specification indicates whether or not a column can contain null values.

To create a table using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click the **Tables** folder, and select **Create —> Tables Using Wizard** from the pop-up menu.
3. Follow the steps in the wizard to complete your tasks.

To create a table using the command line, enter:

```
CREATE TABLE <NAME>
   (<column_name>  <data_type>  <null_attribute>)
   IN <TABLE_SPACE_NAME)
```

The following is an example of a CREATE TABLE statement that creates the EMPLOYEE table in the RESOURCE table space. This table is defined in the sample database:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12) NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15) NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL)
IN RESOURCE
```

When creating a table, you can choose to have the columns of the table based on the attributes of a structured type. Such a table is called a “typed table”.

A typed table can be defined to inherit some of its columns from another typed table. Such a table is called a “subtable”, and the table from which it inherits is called its “supertable”. The combination of a typed table and all its subtables is called a “table hierarchy”. The topmost table in the table hierarchy (the one with no supertable) is called the “root table” of the hierarchy.

The following sections build on the previous example to cover other options you should consider:

- “Large Object (LOB) Column Considerations” on page 121

- "Defining a Unique Constraint" on page 123
- "Defining a Generated Column on a New Table" on page 127
- "Creating a User-Defined Temporary Table" on page 129
- "Defining an Identity Column on a New Table" on page 130
- "Creating a Sequence" on page 131
- "Comparing IDENTITY Columns and Sequences" on page 133
- "Creating a Typed Table" on page 133
- "Populating a Typed Table" on page 133
- "Hierarchy Table" on page 134
- "Creating a Table in Multiple Table Spaces" on page 134
- "Creating a Table in a Partitioned Database" on page 135

You can also create a table that is defined based on the result of a query. This type of table is called a *summary table*. For more information, see "Creating a Summary Table" on page 147.

Large Object (LOB) Column Considerations

Before creating a table that contains large object columns, you need to make the following decisions:

1. Do you want to log changes to LOB columns?

   If you do not want to log these changes, you must turn logging off by specifying the NOT LOGGED clause when you create the table:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12) NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15) NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED)
IN RESOURCE
```

   If the LOB column is larger than 1 GB, logging must be turned off. (As a rule of thumb, you may not want to log LOB columns larger than 10 MB.) As with other options specified on a column definition, the only way to change the logging option is to re-create the table.

   Even if you choose not to log changes, LOB columns are *shadowed* to allow changes to be rolled back, whether the roll back is the result of a system generated error, or an application request. Shadowing is a recovery technique where current storage page contents are never overwritten. That is, old, unmodified pages are kept as "shadow" copies. These copies are discarded when they are no longer needed to support a transaction rollback.

Note: When recovering a database using the RESTORE and ROLLFORWARD commands, LOB data that was "NOT LOGGED"and was written since the last backup will be *replaced by binary zeros*.

2. Do you want to minimize the space required for the LOB column?

   You can make the LOB column as small as possible using the COMPACT clause on the CREATE TABLE statement. For example:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12) NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15) NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED  COMPACT)
IN RESOURCE
```

   There is a *performance cost* when appending to a table with a compact LOB column, particularly if the size of LOB values are increased (because of storage adjustments that must be made).

   On platforms such as OS/2 where sparse file allocation is not supported and where LOBs are placed in SMS table spaces, consider using the COMPACT clause. Sparse file allocation has to do with how physical disk space is used by an operating system. An operating system that supports sparse file allocation does not use as much physical disk space to store LOBs as compared to an operating system not supporting sparse file allocation. The COMPACT option allows for even greater physical disk space "savings" regardless of the support of sparse file allocation. Because you can get some physical disk space savings when using COMPACT, you should consider using COMPACT if your operating system does not support sparse file allocation.

   Note: DB2 system catalogs use LOB columns and may take up more space than in previous versions.

3. Do you want better performance for LOB columns, including those LOB columns in the DB2 system catalogs?

   There are large object (LOB) columns in the catalog tables. LOB data is not kept in the buffer pool with other data but is read from disk each time it is needed. Reading from disk slows down the performance of DB2 where the LOB columns of the catalogs are involved. Since a file system usually has its own place for storing (or caching) data, using a SMS table space, or a DMS table space built on file containers, make avoidance of I/O possible when the LOB has previously been referenced.

Defining Constraints

This section discusses how to define constraints:

- "Defining a Unique Constraint"
- "Defining Referential Constraints" on page 124
- "Defining a Table Check Constraint" on page 127.

For more information on constraints, refer to the section on planning for constraint enforcement in the *Administration Guide: Planning*; and to the *SQL Reference*.

Defining a Unique Constraint

Unique constraints ensure that every value in the specified key is unique. A table can have any number of unique constraints, with at most one unique constraint defined as a primary key.

You define a unique constraint with the UNIQUE clause in the CREATE TABLE or ALTER TABLE statements. The unique key can consist of more than one column. More than one unique constraint is allowed on a table. However, a unique constraint may not be defined on a subtable.

Once established, the unique constraint is enforced automatically by the database manager when an INSERT or UPDATE statement modifies the data in the table. The unique constraint is enforced through a unique index.

When a unique constraint is defined in an ALTER TABLE statement and an index exists on the same set of columns of that unique key, that index becomes the unique index and is used by the constraint.

You can take any one unique constraint and use it as the *primary key*. The primary key can be used as the parent key in a referential constraint (along with other unique constraints). There can be only one primary key per table. You define a primary key with the PRIMARY KEY clause in the CREATE TABLE or ALTER TABLE statement. The primary key can consist of more than one column.

A primary index forces the value of the primary key to be unique. When a table is created with a primary key, the database manager creates a primary index on that key.

Some performance tips for indexes used as unique constraints include:

- When performing an initial load of an empty table with indexes, LOAD gives better performance than IMPORT. This is true no matter whether you are using the INSERT or REPLACE modes of LOAD.

- When appending a substantial amount of data to an existing table with indexes (using IMPORT INSERT, or LOAD INSERT), LOAD gives slightly better performance than IMPORT.
- If you are using the IMPORT command for an initial large load of data, create the unique key after the data has been imported or loaded. This avoids the overhead of maintaining the index while the table is being loaded. It also results in the index using the least amount of storage.
- If you are using the load utility in REPLACE mode, create the unique key before loading the data. In this case, creation of the index during the load is more efficient than using the CREATE INDEX statement after the load.

Defining Referential Constraints

Referential integrity is imposed by adding referential constraints to table and column definitions. Referential constraints are established with the FOREIGN KEY clause, and the REFERENCES clause in the CREATE TABLE or ALTER TABLE statements. Refer to the *SQL Reference* for more information on the effects of a referential constraint on typed tables or to a parent table that is a typed table.

The identification of foreign keys enforces constraints on the values within the rows of a table or between the rows of two tables. The database manager checks the constraints specified in a table definition and maintains the relationships accordingly. The goal is to maintain integrity whenever one database object references another.

For example, primary and foreign keys each have a department number column. For the EMPLOYEE table, the column name is WORKDEPT, and for the DEPARTMENT table, the name is DEPTNO. The relationship between these two tables is defined by the following constraints:

- There is only one department number for each employee in the EMPLOYEE table, and that number exists in the DEPARTMENT table.
- Each row in the EMPLOYEE table is related to no more than one row in the DEPARTMENT table. There is a unique relationship between the tables.
- Each row in the EMPLOYEE table that has a non-null value for WORKDEPT is related to a row in the DEPTNO column of the DEPARTMENT table.
- The DEPARTMENT table is the parent table, and the EMPLOYEE table is the dependent table.

The SQL statement defining the parent table, DEPARTMENT, is:

```
CREATE TABLE DEPARTMENT
   (DEPTNO    CHAR(3)     NOT NULL,
    DEPTNAME  VARCHAR(29) NOT NULL,
    MGRNO     CHAR(6),
```

```
    ADMRDEPT  CHAR(3)     NOT NULL,
    LOCATION  CHAR(16),
       PRIMARY KEY (DEPTNO))
IN RESOURCE
```

The SQL statement defining the dependent table, EMPLOYEE, is:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12) NOT NULL,
    LASTNAME  VARCHAR(15) NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10m)   NOT NULL,
       FOREIGN KEY DEPT (WORKDEPT)
       REFERENCES DEPARTMENT ON DELETE NO ACTION)
IN RESOURCE
```

By specifying the DEPTNO column as the primary key of the DEPARTMENT table and WORKDEPT as the foreign key of the EMPLOYEE table, you are defining a referential constraint on the WORKDEPT values. This constraint enforces referential integrity between the values of the two tables. In this case, any employees that are added to the EMPLOYEE table must have a department number that can be found in the DEPARTMENT table.

The delete rule for the referential constraint in the employee table is NO ACTION, which means that a department cannot be deleted from the DEPARTMENT table if there are any employees in that department.

Although the previous examples use the CREATE TABLE statement to add a referential constraint, the ALTER TABLE statement can also be used. See “Modifying a Table in Both Structure and Content” on page 188.

Another example: The same table definitions are used as those in the previous example. Also, the DEPARTMENT table is created before the EMPLOYEE table. Each department has a manager, and that manager is listed in the EMPLOYEE table. MGRNO of the DEPARTMENT table is actually a foreign key of the EMPLOYEE table. Because of this referential cycle, this constraint poses a slight problem. You could add a foreign key later (see “Adding Primary and Foreign Keys” on page 192). You could also use the CREATE SCHEMA statement to create both the EMPLOYEE and DEPARTMENT tables at the same time (see the example in the *SQL Reference*).

FOREIGN KEY Clause: A foreign key references a primary key or a unique key in the same or another table. A foreign key assignment indicates that referential integrity is to be maintained according to the specified referential constraints. You define a foreign key with the FOREIGN KEY clause in the CREATE TABLE or ALTER TABLE statement.

The number of columns in the foreign key must be equal to the number of columns in the corresponding primary or unique constraint (called a parent key) of the parent table. In addition, corresponding parts of the key column definitions must have the same data types and lengths. The foreign key can be assigned a *constraint name*. If you do not assign a name, one is automatically assigned. For ease of use, it is recommended that you assign a *constraint name* and do not use the system-generated name.

The value of a composite foreign key matches the value of a parent key **if** the value of each column of the foreign key is equal to the value of the corresponding column of the parent key. A foreign key containing null values cannot match the values of a parent key, since a parent key by definition can have no null values. However, a null foreign key value is always valid, regardless of the value of any of its non-null parts.

The following rules apply to foreign key definitions:

- A table can have many foreign keys
- A foreign key is nullable if any part is nullable
- A foreign key value is null if any part is null.

REFERENCES Clause: The REFERENCES clause identifies the parent table in a relationship, and defines the necessary constraints. You can include it in a column definition or as a separate clause accompanying the FOREIGN KEY clause, in either the CREATE TABLE or ALTER TABLE statements.

If you specify the REFERENCES clause as a column constraint, an implicit column list is composed of the column name or names that are listed. Remember that multiple columns can have separate REFERENCES clauses, and that a single column can have more than one.

Included in the REFERENCES clause is the delete rule. In our example, the ON DELETE NO ACTION rule is used, which states that no department can be deleted if there are employees assigned to it. Other delete rules include ON DELETE CASCADE, ON DELETE SET NULL, and ON DELETE RESTRICT. Refer to *Administration Guide: Planning* for more information on DELETE rules when implementing referential integrity.

Implications for Utility Operations: The load utility will turn off constraint checking for self-referencing and dependent tables, placing these tables into check pending state. After the load utility has completed, you will need to turn on the constraint checking for all tables for which it was turned off. For example, if the DEPARTMENT and EMPLOYEE tables are the only tables that have been placed in check pending state, you can execute the following command:

```
SET INTEGRITY FOR DEPARTMENT, EMPLOYEE IMMEDIATE CHECKED
```

The import utility is affected by referential constraints in the following ways:

- The REPLACE and REPLACE CREATE functions are not allowed if the object table has any dependents other than itself.

  To use these functions, first drop all foreign keys in which the table is a parent. When the import is complete, re-create the foreign keys with the ALTER TABLE statement.
- The success of importing into a table with self-referencing constraints depends on the order in which the rows are imported.

Defining a Table Check Constraint

A table check constraint specifies a search condition that is enforced for each row of the table on which the table check constraint is defined. You create a table check constraint on a table by associating a check-constraint definition with the table when the table is created or altered. This constraint is automatically activated when an INSERT or UPDATE statement modifies the data in the table. A table check constraint has no effect on a DELETE or SELECT statement. A check constraint can be associated with a typed table.

A constraint name cannot be the same as any other constraint specified within the same CREATE TABLE statement. If you do not specify a constraint name, the system generates an 18-character unique identifier for the constraint.

A table check constraint is used to enforce data integrity rules not covered by key uniqueness or a referential integrity constraint. In some cases, a table check constraint can be used to implement domain checking. The following constraint issued on the CREATE TABLE statement ensures that the start date for every activity is not after the end date for the same activity:

```
CREATE TABLE EMP_ACT
   (EMPNO      CHAR(6)      NOT NULL,
    PROJNO     CHAR(6)      NOT NULL,
    ACTNO      SMALLINT     NOT NULL,
    EMPTIME    DECIMAL(5,2),
    EMSTDATE   DATE,
    EMENDATE   DATE,
    CONSTRAINT ACTDATES CHECK(EMSTDATE <= EMENDATE) )
IN RESOURCE
```

Although the previous example uses the CREATE TABLE statement to add a table check constraint, the ALTER TABLE statement can also be used. See "Modifying a Table in Both Structure and Content" on page 188.

Defining a Generated Column on a New Table

A generated column is defined in a base table where the stored value is computed using an expression, rather than being specified through an insert or update operation. When creating a table where it is known that certain expressions or predicates will be used all the time, you can add one or more

generated columns to that table. By using a generated column there is opportunity for performance improvements when querying the table data.

For example, there are two ways in which the evaluation of expressions can be costly when performance is important:

1. The evaluation of the expression must be done many times during a query.
2. The computation is complex.

To improve the performance of the query, you can define an additional column that would contain the results of the expression. Then, when issuing a query that includes the same expression, the generated column can be used directly; or, the query rewrite component of the optimizer can replace the expression with the generated column.

It is also possible to create a non-unique index on a generated column.

Where queries involve the joining of data from two or more tables, the addition of a generated column can allow the optimizer a choice of possibly better join strategies.

The following is an example of defining a generated column on the CREATE TABLE statement:

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

After creating this table, indexes can be created using the generated columns. For example,

```
CREATE INDEX i1 ON t1(c4)
```

Queries can take advantage of the generated columns. For example,

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

can be written as

```
SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

Another example:

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

can be written as

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

Generated columns will be used to improve performance of queries. As a result, generated columns will likely be added after the table has been created and populated. See "Creating and Populating a Table" on page 118 for more information.

Creating a User-Defined Temporary Table

You use the DECLARE GLOBAL TEMPORARY TABLE statement to define a temporary table. The statement is used from within an application. The user-defined temporary table only persists until the application disconnects from the database.

The description of this table does not appear in the system catalog making it not persistent for, and not able to be shared with, other applications.

When the application using this table terminates or disconnects from the database, any data in the table is deleted and the table is implicitly dropped.

An example of how you can define a temporary table as follows:

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

This statement creates a user temporary table called `gbl_temp`. The user temporary table is defined with columns that have exactly the same name and description as the columns of the `empltabl`. 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. When a COMMIT operation is performed, all data in the table is deleted if no WITH HOLD cursor is open on the table. Changes made to the user temporary table are not logged. The user temporary table is placed in the specified user temporary table space. This table space must exist or the declaration of this table will fail.

Refer to the *SQL Reference* for additional information on the DECLARE GLOBAL TEMPORARY TABLE statement.

Note: A user-defined temporary table does not support:

- LOB-type columns (or a distinct-type column based on a LOB)
- User-defined type columns
- LONG VARCHAR columns
- DATALINK columns

Defining an Identity Column on a New Table

An *identity column* provides a way for DB2 to automatically generate a guaranteed-unique numeric value for each row that is added to the table. When creating a table where you know that you need to uniquely identify each row that will be added to the table, you can add an identity column to the table.

Once created, you cannot alter the table description to include an identity column.

It is the AS IDENTITY clause on the CREATE TABLE statement that allows for the specification of the identity column.

The following is an example of defining an identity column on the CREATE TABLE statement:

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

In this example the third column is the identity column. You can also specify the value used in the column to uniquely identify each row when added. Here the first row entered has the value of "100" placed in the column; every subsequent row added to the table has the associated value increased by five.

Some additional example uses of an identity column are an order number, an employee number, a stock number, or an incident number. The values for an identity column can be generated by DB2: ALWAYS or BY DEFAULT.

An identity column defined as GENERATED ALWAYS is guaranteed to be unique. The values used are always generated by DB2. Applications are not allowed to provide an explicit value. An identity column defined as GENERATED BY DEFAULT gives applications a way to explicitly provide a value for the identity column. If the application does not provide a value, then DB2 will generate one. Since the application controls the value, DB2 cannot guarantee the uniqueness of the value. The GENERATED BY DEFAULT clause is meant for use for data propagation where the intent is to copy the contents of an existing table; or, for the unload and reloading of a table.

Note: Identity columns are not currently supported in a partitioned database environment.

If rows are inserted into a table with explicit identity column values specified, the next internally generated value is not updated, and may conflict with existing values in the table. Duplicate values will generate an error message.

Refer to *SQL Reference* for additional information on defining an identity column on a new table.

Creating a Sequence

A *sequence* is a database object that allows the automatic generation of values. Sequences are ideally suited to the task of generating unique key values. Applications can use sequences to avoid possible concurrency and performance problems resulting from the generation of a unique counter outside the database.

Unlike an identity column attribute, a sequence is not tied to a particular table column nor is it bound to a unique table column and only accessible through that table column.

A sequence can be created, or altered, so that it generates values in one of these ways:

- Increment or decrement monotonically without bound
- Increment or decrement monotonically to a user-defined limit and stop
- Increment or decrement monotonically to a user-defined limit and cycle back to the beginning and start again

The following is an example of creating a sequence object:

```
CREATE SEQUENCE order_seq
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE
   CACHE 24
```

In this example, the sequence is called `order_seq`. It will start at 1 and increase by 1 with no upper limit. There is no reason to cycle back to the beginning and restart from 1 because there is no assigned upper limit. The number associated with the `CACHE` parameter specifies the maximum number of sequence values that the database manager preallocates and keeps in memory.

The sequence numbers generated have the following properties:

- Values can be any exact numeric data type with a scale of zero. Such data types include: SMALLINT, BIGINT, INTEGER, and DECIMAL.
- Consecutive values can differ by any specified integer increment. The default increment value is 1.
- Counter value is recoverable. The counter value is reconstructed from logs when recovery is required.
- Values can be cached to improve performance. Preallocating and storing values in the cache reduces synchronous I/O to the log when values are

generated for the sequence. In the event of a system failure, all cached values that have not been committed are never used and considered lost. The value specified for CACHE is the maximum number of sequence values that could be lost.

If a database that contains one or more sequences is recovered to a prior point in time, then this could cause the generation of duplicate values for some sequences. To avoid possible duplicate values, a database with sequences should not be recovered to a prior point in time.

Sequences are only supported in a single node database.

There are two expressions used with a sequence.

The PREVVAL expression returns the most recently generated value for the specified sequence for a previous statement within the current session.

The NEXTVAL expression returns the next value for the specified sequence. A new sequence number is generated when a NEXTVAL expression specifies the name of the sequence. However, if there are multiple instances of a NEXTVAL expression specifying the same sequence name within a query, the counter for the sequence is incremented only once for each row of the result.

The same sequence number can be used as a unique key value in two separate tables by referencing the sequence number with a NEXTVAL expression for the first table, and a PREVVAL expression for any additional tables.

For example:

```
INSERT INTO order (orderno, custno)
   VALUES (NEXTVAL FOR order_seq, 123456);
INSERT INTO line_item (orderno, partno, quantity)
   VALUES (PREVVAL FOR order_seq, 987654, 1)
```

The NEXTVAL or PREVVAL expressions can be used in the following locations:

- INSERT statement, VALUES clause
- SELECT statement, SELECT list
- SET assignment statement
- UPDATE statement, SET clause
- VALUES or VALUES INTO statement

Comparing IDENTITY Columns and Sequences

While there are similarities between IDENTITY columns and sequences, there are also differences. The characteristics of each can be used when designing your database and applications.

An identity column has the following characteristics:

- An identity column can be defined as part of a table only when the table is created. Once a table is created, you cannot alter it to add an identity column. (However, existing identity column characteristics may be altered.)
- An identity column automatically generates values for a single table.
- When an identity column is defined as GENERATED ALWAYS, the values used are always generated by the database manager. Applications are not allowed to provide their own values during the modification of the contents of the table.

A sequence object has the following characteristics:

- A sequence object is a database object that is not tied to any one table.
- A sequence object generates sequential values that can be used in any SQL statement.
- Since a sequence object can be used by any application, there are two expressions used to control the retrieval of the next value in the specified sequence and the value generated previous to the statement being executed. The PREVVAL expression returns the most recently generated value for the specified sequence for a previous statement within the current session. The NEXTVAL expression returns the next value for the specified sequence. The use of these expressions allows the same value to be used across several SQL statements within several tables.

While these are not all of the characteristics of these two items, these characteristics will assist you in determining which to use depending on your database design and the applications using the database.

Creating a Typed Table

You can create a typed table using a variant of the CREATE TABLE statement. Refer to the *Application Development Guide* for all the information you need on typed tables.

Populating a Typed Table

You can populate a typed table after creating the structured types and then creating the corresponding tables and subtables. Refer to the *Application Development Guide* for all the information you need on typed tables.

Hierarchy Table

A hierarchy table is a table that is associated with the implementation of a typed table hierarchy. It is created at the same time as the root table of the hierarchy. Refer to the *Application Development Guide* for all the information you need on hierarchy tables.

Creating a Table in Multiple Table Spaces

Table data can be stored in the same table space as the index for the table, and any long column data associated with the table. You can also place the index in a separate table space, and place any long column data in a separate table space, apart from the table space for the rest of the table data. All table spaces must exist before the CREATE TABLE statement is run. The separation of the parts of the table can only be done using DMS table spaces.

To create a table in multiple table spaces using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click the **Tables** folder, and select **Create —> Tables Using Wizard** from the pop-up menu.
3. Type the table name and click **Next**.
4. Select columns for your table.
5. On the **Table space** page, click **Use separate index space** and **Use separate long space**, specify the information, and click **Finish**.

To create a table in multiple table spaces using the command line, enter:

```
CREATE TABLE <name>
   (<column_name>  <data_type>  <null_attribute>)
    IN <table_space_name>
    INDEX IN <index_space_name>
    LONG  IN <long_space_name>
```

The following example shows how the EMP_PHOTO table could be created to store the different parts of the table in different table spaces:

```
CREATE TABLE EMP_PHOTO
   (EMPNO        CHAR(6)      NOT NULL,
    PHOTO_FORMAT VARCHAR(10)  NOT NULL,
    PICTURE      BLOB(100K) )
IN RESOURCE
INDEX IN RESOURCE_INDEXES
LONG  IN RESOURCE_PHOTO
```

This example will cause the EMP_PHOTO data to be stored as follows:

- Indexes created for the EMP_PHOTO table will be stored in the RESOURCES_INDEXES table space

- Data for the PICTURE column will be stored in the RESOURCE_PHOTO table space
- Data for the EMPNO and PHOTO_FORMAT columns will be stored in the RESOURCE table space.

For additional considerations on the use of multiple DMS table spaces for a single table, refer to *Administration Guide: Planning*.

Refer to the *SQL Reference* for more information about the CREATE TABLE statement.

Creating a Table in a Partitioned Database

Before creating a table that will be physically divided or partitioned, you need to consider the following:

- Table spaces can span more than one database partition. The number of partitions they span depends on the number of partitions in a nodegroup.
- Tables can be collocated by being placed in the same table space or by being placed in another table space that, together with the first table space, is associated with the same nodegroup. For more information, refer to *Administration Guide: Planning*.

One additional option exists when creating a table in a partitioned database environment: the *partitioning key*. A partitioning key is a key that is part of the definition of a table. It determines the partition on which each row of data is stored.

It is important to select an appropriate partitioning key because *it cannot be changed later*. Furthermore, any unique indexes (and therefore unique or primary keys) must be defined as a superset of the partitioning key. That is, if a partitioning key is defined, unique keys and primary keys must include all of the same columns as the partitioning key (they may have more columns).

If you do not specify the partitioning key explicitly, the following defaults are used. *Ensure that the default partitioning key is appropriate.*

- If a primary key is specified in the CREATE TABLE statement, the first column of the primary key is used as the partitioning key.
- If there is no primary key, the first column that is not a long field is used.
- If no columns satisfy the requirements for a default partitioning key, the table is created without one (this is allowed only in single-partition nodegroups).

Following is an example:

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
```

```
MIX_INT INTEGER NOT NULL,
MIX_INTS SMALLINT NOT NULL,
MIX_DEC DECIMAL NOT NULL,
MIX_FLT FLOAT NOT NULL,
MIX_DATE DATE NOT NULL,
MIX_TIME TIME NOT NULL,
MIX_TMSTMP TIMESTAMP NOT NULL)
IN MIXTS12
PARTITIONING KEY (MIX_INT) USING HASHING
```

In the preceding example, the table space is MIXTS12 and the partitioning key is MIX_INT. If the partitioning key is not specified explicitly, it is MIX_CNTL. (If no primary key is specified and no partitioning key is defined, the partitioning key is the first non-long column in the list.)

A row of a table, and all information about that row, always resides on the same database partition.

The size limit for one partition of a table is 64 GB, or the available disk space, whichever is smaller. (This assumes a 4 KB page size for the table space.) The size of the table can be as large as 64 GB (or the available disk space) times the number of database partitions. If the page size for the table space is 8 KB, the size of the table can be as large as 128 GB (or the available disk space) times the number of database partitions. If the page size for the table space is 16 KB, the size of the table can be as large as 256 GB (or the available disk space) times the number of database partitions. If the page size for the table space is 32 KB, the size of the table can be as large as 512 GB (or the available disk space) times the number of database partitions.

Creating a Trigger

A trigger defines a set of actions that are executed in conjunction with, or triggered by, an INSERT, UPDATE, or DELETE clause on a specified base table or a typed table. Some uses of triggers are to:

- Validate input data
- Generate a value for a newly-inserted row
- Read from other tables for cross-referencing purposes
- Write to other tables for audit-trail purposes

You cannot use triggers with nicknames.

You can use triggers to support general forms of integrity or business rules. For example, a trigger can check a customer's credit limit before an order is accepted or update a summary data table.

The benefits of using a trigger are:

- Faster application development: Because a trigger is stored in the database, you do not have to code the actions it does in every application.
- Easier maintenance: Once a trigger is defined, it is automatically invoked when the table that it is created on is accessed.
- Global enforcement of business rules: If a business policy changes, you only need to change the trigger and not each application program.

To create a trigger using the Control Center:

1. Expand the object tree until you see the **Triggers** folder.
2. Right-click the **Triggers** folder, and select **Create** from the pop-up menu.
3. Specify information for the trigger.
4. Specify the action that you want the trigger to invoke, and click **Ok**.

To create a trigger using the command line, enter:

```
CREATE TRIGGER <name>
   <action> ON <table_name>
   <operation>
   <triggered_action>
```

The following SQL statement creates a trigger that increases the number of employees each time a new person is hired, by adding 1 to the number of employees (NBEMP) column in the COMPANY_STATS table each time a row is added to the EMPLOYEE table.

```
CREATE TRIGGER NEW_HIRED
   AFTER INSERT ON EMPLOYEE
   FOR EACH ROW MODE DB2SQL
   UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

A trigger body can include one or more of the following SQL statements: INSERT, searched UPDATE, searched DELETE, full-selects, SET transition-variable, and SIGNAL SQLSTATE. The trigger can be activated before or after the INSERT, UPDATE, or DELETE statement to which it refers. Refer to the *SQL Reference* for complete syntax information on the CREATE TRIGGER statement. Refer to the *Application Development Guide* for information about creating and using triggers.

Note: If the trigger is a BEFORE trigger, the column name specified by the triggered action may not be a generated column other than an identity column. That is, the generated identity value is visible to BEFORE triggers.

When creating an atomic trigger, care must be taken with the end-of-statement character. The database manager, by default, considers a ";" the end-of-statement marker. You should manually edit the end-of-statement

character in your script to create the atomic trigger so that you are using a character other than ";". For example, the ";" could be replaced by another special character like "#".

Then you must either:

- Change the delimiter from the tools—>tools settings menu with script tab selected in the Command Center and then run the script; Or,
- From the Command Line Processor, use:

```
db2 -td <delimiter> -vf <script>
```

  where the delimiter is the alternative end-of-statement character and the <script> is the modified script with the new delimiter in it.

Trigger Dependencies

All dependencies of a trigger on some other object are recorded in the SYSCAT.TRIGDEP catalog. A trigger can depend on many objects. These objects and the dependent trigger are presented in detail in the *SQL Reference* discussion on the DROP statement.

If one of these objects is dropped, the trigger becomes inoperative but its definition is retained in the catalog. To revalidate this trigger, you must retrieve its definition from the catalog and submit a new CREATE TRIGGER statement.

If a trigger is dropped, its description is deleted from the SYSCAT.TRIGGERS catalog view and all of its dependencies are deleted from the SYSCAT.TRIGDEP catalog view. All packages having UPDATE, INSERT, or DELETE dependencies on the trigger are invalidated.

If the dependent object is a view and it is made inoperative, the trigger is also marked inoperative. Any packages dependent on triggers that have been marked inoperative are invalidated. (For more information, see "Statement Dependencies When Changing Objects" on page 215.)

Creating a User-Defined Function (UDF) or Method

User-defined functions (UDFs) extend and add to the support provided by built-in functions of SQL, and can be used wherever a built-in function can be used. You can create UDFs as either:

- An external function, which is written in a programming language
- A sourced function, whose implementation is inherited from some other existing function

There are three types of UDFs:

Scalar Returns a single-valued answer each time it is called. For example, the built-in function SUBSTR() is a scalar function. Scalar UDFs can be either external or sourced.

Column

Returns a single-valued answer from a set of like values (a column). It is also sometimes called an aggregating function in DB2. An example of a column function is the built-in function AVG(). An external column UDF cannot be defined to DB2, but a column UDF which is sourced upon one of the built-in column functions can be defined. This is useful for distinct types.

For example, if there is a distinct type SHOESIZE defined with base type INTEGER, a UDF AVG(SHOESIZE) which is sourced on the built-in function AVG(INTEGER) could be defined, and it would be a column function.

Table Returns a table to the SQL statement which references it. Table functions may only be referenced in the FROM clause of a SELECT statement. Such a function can be used to apply SQL language processing power to data which is not DB2 data, or to convert such data into a DB2 table.

For example, table functions can take a file and convert it to a table, tabularize sample data from the World Wide Web, or access a Lotus Notes database and return information such as the date, sender, and text of mail messages. This information can be joined with other tables in the database.

A table function can only be an external function. It cannot be a sourced function.

Information about existing UDFs is recorded in the SYSCAT.FUNCTIONS and SYSCAT.FUNCPARMS catalog views. The system catalog does *not* contain the executable code for the UDF. (Therefore, when creating your backup and recovery plans you should consider how you will manage your UDF executables.)

Statistics about the performance of UDFs are important when compiling SQL statements. For information about how to update UDF statistics in the system catalog, refer to "Updating Statistics for User-Defined Functions" in *Administration Guide: Performance*.

For details on using the CREATE FUNCTION statement to write a UDF to suit your specific application, refer to the *Application Development Guide*. Refer to the *SQL Reference* for details on UDF syntax.

Creating a Function Mapping

In a federated database, create a function mapping when you need to map a local function or a local function template (described in "Creating a Function Template") with a function at one or more data sources. Default function mappings are provided for many data source functions.

Function mappings are useful when:

- New, built-in functions become available at a data source.
- You need to map a user-defined function at a data source to a local function.
- An application requires different default behavior than that provided by the default mapping.

Function mappings defined with CREATE FUNCTION MAPPING statements are stored in the federated database.

Functions (or function templates) must have the same number of input parameters as the data source function. Additionally, the data types of the input parameters on the federated side should be compatible with the data types of the input parameters on the data source side. These requirements apply to returned values as well.

Use the CREATE FUNCTION MAPPING statement to create a function mapping. For example, to create a function mapping between an Oracle AVGNEW function and a DB2 equivalent at server ORACLE1:

```
CREATE FUNCTION MAPPING ORAVGNEW FOR SYSIBM.AVG(INT) SERVER ORACLE1
OPTIONS (REMOTE_NAME 'AVGNEW')
```

You must hold one of the SYSADM or DBADM authorities at the federated database to use this statement. Function mapping attributes are stored in SYSCAT.FUNCMAPPINGS.

The federated server will not bind input host variables or retrieve results of LOB, LONG VARCHAR/VARGRAPHIC, DATALINK, distinct and structured types. No function mapping can be created when an input parameter or the returned value includes one of these types.

For additional details on using and creating function mappings, refer to the *Application Development Guide*. Refer to the *SQL Reference* for details on CREATE FUNCTION MAPPING syntax.

Creating a Function Template

In a federated system, function templates provide "anchors" for function mappings. They are used to enable the mapping of a data source function

when a corresponding DB2 function does not exist at the federated server. A function mapping requires the presence of a function template or an existing similar function at DB2.

The template is just a function shell: name, input parameters, and the return value. There is no local executable for the function.

Because there is no local executable for the function, it is possible that a call to the function template will fail even though the function is available at the data source. For example, consider the query:

```
SELECT myfunc(C1)
   FROM nick1
   WHERE C2 < 'A'
```

If DB2 and the data source containing the object referenced by nick1 do not have the same collating sequence, the query will fail because the comparison must be done at DB2 while the function is at the data source. If the collating sequences were the same, the comparison operation could be done at the data source that has the underlying function referenced by `myfunc`.

Functions (or function templates) must have the same number of input parameters as the data source function. The data types of the input parameters on the federated side should be compatible with the data types of the input parameters on the data source side. These requirements apply to returned values as well.

You create function templates using the CREATE FUNCTION statement with the AS TEMPLATE keyword. After the template is created, you map the template to the data source using the CREATE FUNCTION MAPPING statement.

For example, to create a function template and a function mapping for function MYS1FUNC on server S1:

```
CREATE FUNCTION MYFUNC(INT) RETURNS INT AS TEMPLATE

CREATE FUNCTION MAPPING S1_MYFUNC FOR MYFUNC(INT) SERVER S1 OPTIONS
(REMOTE_NAME 'MYS1FUNC')
```

For details on using and creating function templates, refer to the *Application Development Guide*. Refer to the *SQL Reference* for details on CREATE FUNCTION syntax.

Creating a User-Defined Type (UDT)

A user-defined type (UDT) is a named data type that is created in the database by the user. A UDT can be a distinct type which shares a common representation with a built-in data type or a structured type which has a sequence of named attributes that each have a type. A structured type can be a subtype of another structured type (called a supertype), defining a type hierarchy.

UDTs support strong typing, which means that even though they share the same representation as other types, values of a given UDT are considered to be compatible only with values of the same UDT or UDTs in the same type hierarchy.

The SYSCAT.DATATYPES catalog view allows you to see the UDTs that have been defined for your database. This catalog view also shows you the data types defined by the database manager when the database was created. For a complete list of all data types, refer to the *SQL Reference*.

A UDT cannot be used as an argument for most of the system-provided, or built-in, functions. User-defined functions must be provided to enable these and other operations.

You can drop a UDT only if:

- It is *not* used in a column definition for an existing table.
- It is *not* used as the type of an existing typed table or typed view.
- It is *not* used in a UDF function that cannot be dropped. A UDF cannot be dropped if a view, trigger, table check constraint, or another UDF is dependent on it.

When a UDT is dropped, any functions that are dependent on it are also dropped.

Creating a User-Defined Distinct Type

A user-defined distinct type is a data type derived from an existing type, such as an integer, decimal, or character type. You can create a distinct type by using the CREATE DISTINCT TYPE statement.

The following SQL statement creates the distinct type `t_educ` as a smallint:

```
CREATE DISTINCT TYPE T_EDUC AS SMALLINT WITH COMPARISONS
```

Instances of the same distinct type can be compared to each other, if the WITH COMPARISONS clause is specified on the CREATE DISTINCT TYPE statement (as in the example). The WITH COMPARISONS clause cannot be specified if the source data type is a large object, a DATALINK, LONG VARCHAR, or LONG VARGRAPHIC type.

Instances of distinct types cannot be used as arguments of functions or operands of operations that were defined on the source type. Similarly, the source type cannot be used in arguments or operands that were defined to use a distinct type.

After you have created a distinct type, you can use it to define columns in a CREATE TABLE statement:

```
CREATE TABLE EMPLOYEE
   (EMPNO      CHAR(6)      NOT NULL,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO      BLOB(10M)    NOT NULL,
    EDLEVEL    T_EDUC)
IN RESOURCE
```

Creating the distinct type also generates support to cast between the distinct type and the source type. Hence, a value of type T_EDUC can be cast to a SMALLINT value and the SMALLINT value can be cast to a T_EDUC value.

Refer to the *SQL Reference* for complete syntax information on the CREATE DISTINCT TYPE statement. Refer to the *Application Development Guide* for information about creating and using a distinct type.

You can transform UDTs into base data types, and base data types into UDTs, using transformations. Creation of a transform function is through a CREATE TRANSFORM statement.

Support for transforms is also found through the CREATE METHOD statement and extensions to the CREATE FUNCTION statement. Refer to the *SQL Reference* for details on this support.

Creating a User-Defined Structured Type

A structured type is a user-defined type that contains one or more attributes, each of which has a name and a data type of its own. A structured type can serve as the type of a table, in which each column of the table derives its name and data type from one of the attributes of the structured type. Refer to the *Application Development Guide* for all the information you need on structured types.

Creating a Type Mapping

In a federated system, a type mapping lets you map specific data types in data source tables and views to DB2 distinct data types. A type mapping can apply to one data source or a range (type, version) of data sources.

Default data type mappings are provided for built-in data source types and built-in DB2 types. New data type mappings (that you create) will be listed in the SYSCAT.TYPEMAPPINGS view.

You create type mappings with the CREATE TYPE MAPPING statement. You must hold one of the SYSADM or DBADM authorities at the federated database to use this statement.

An example of a type mapping statement is:

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL(10,2)
TO SERVER ORACLE1 TYPE NUMBER([10..38],2)
```

You cannot create a type mapping for a LOB, LONG VARCHAR/VARGRAPHIC, DATALINK, structured or distinct type.

For details on using and creating type mappings, refer to the *Application Development Guide*. Refer to the *SQL Reference* for details on CREATE TYPE MAPPING syntax.

Creating a View

Views are derived from one or more base tables, nicknames, or views, and can be used interchangeably with base tables when retrieving data. When changes are made to the data shown in a view, the data is changed in the table itself.

A view can be created to limit access to sensitive data, while allowing more general access to other data.

When inserting into a view where the SELECT-list of the view definition directly or indirectly includes the name of an identity column of a base table, the same rules apply as if the INSERT statement directly referenced the identity column of the base table. Refer to the *SQL Reference* for more information on the INSERT statement.

In addition to using views as described above, a view can also be used to:

- Alter a table without affecting application programs. This can happen by creating a view based on an underlying table. Applications that use the underlying table are not affected by the creation of the new view. New applications can use the created view for different purposes than those applications that use the underlying table.
- Sum the values in a column, select the maximum values, or average the values.
- Provide access to information in one or more data sources. You can reference nicknames within the CREATE VIEW statement and create

multi-location/global views (the view could join information in multiple data sources located on different systems).

When you create a view that references nicknames using standard CREATE VIEW syntax, you will see a warning alerting you to the fact that the authentication ID of view users will be used to access the underlying object or objects at data sources instead of the view creator authentication ID. Use the FEDERATED keyword to suppress this warning.

An alternative to creating a view is to use a nested or common table expression to reduce catalog lookup and improve performance. Refer to the *SQL Reference* for more information about common table expressions.

To create a view using the Control Center:

1. Expand the object tree until you see the **Views** folder.
2. Right-click the **Views** folder, and select **Create** from the pop-up menu.
3. Complete the information, and click **Ok**.

To create a view using the command line, enter:

```
CREATE VIEW <name> (<column>, <column>, <column>)
   SELECT <column_names> FROM <table_name>
   WITH CHECK OPTION
```

For example, the EMPLOYEE table may have salary information in it, which should not be made available to everyone. The employee's phone number, however, should be generally accessible. In this case, a view could be created from the LASTNAME and PHONENO columns only. Access to the view could be granted to PUBLIC, while access to the entire EMPLOYEE table could be restricted to those who have the authorization to see salary information. For information about *read-only* views, refer to the *SQL Reference* manual.

With a view, you can make a subset of table data available to an application program and validate data that is to be inserted or updated. A view can have column names that are different from the names of corresponding columns in the original tables.

The use of views provides flexibility in the way your programs and end-user queries can look at the table data.

The following SQL statement creates a view on the EMPLOYEE table that lists all employees in Department A00 with their employee and telephone numbers:

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
   AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

The first line of this statement names the view and defines its columns. The name EMP_VIEW must be unique within its schema in SYSCAT.TABLES. The view name appears as a table name although it contains no data. The view will have three columns called DA00NAME, DA00NUM, and PHONENO, which correspond to the columns LASTNAME, EMPNO, and PHONENO from the EMPLOYEE table. The column names listed apply one-to-one to the select list of the SELECT statement. If column names are not specified, the view uses the same names as the columns of the result table of the SELECT statement.

The second line is a SELECT statement that describes which values are to be selected from the database. It may include the clauses ALL, DISTINCT, FROM, WHERE, GROUP BY, and HAVING. The name or names of the data objects from which to select columns for the view must follow the FROM clause.

The WITH CHECK OPTION clause indicates that any updated or inserted row to the view must be checked against the view definition, and rejected if it does not conform. This enhances data integrity but requires additional processing. If this clause is omitted, inserts and updates are not checked against the view definition.

The following SQL statement creates the same view on the EMPLOYEE table using the SELECT AS clause:

```
CREATE VIEW EMP_VIEW
   SELECT LASTNAME AS DA00NAME,
          EMPNO AS DA00NUM,
          PHONENO
   FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

You can create a view that uses a UDF in its definition. However, to update this view so that it contains the latest functions, you must drop it and then re-create it. If a view is dependent on a UDF, that function cannot be dropped.

The following SQL statement creates a view with a function in its definition:

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
  AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
  FROM EMPLOYEE
```

The UDF function PENSION calculates the current pension an employee is eligible to receive, based on a formula involving their HIREDATE, BIRTHDATE, SALARY, and BONUS.

Creating a Typed View

You can create a typed view using the CREATE VIEW statement. Refer to the *Application Development Guide* for all the information you need on typed views.

Creating a Summary Table

A *summary table* is a table whose definition is based on the result of a query. As such, the summary table typically contains pre-computed results based on the data existing in the table or tables that its definition is based on. If the SQL compiler determines that a query will run more efficiently against a summary table than the base table, the query executes against the summary table, and you obtain the result faster than you otherwise would.

The creation of a summary table with the replication option can be used to replicate tables across all nodes in a partitioned database environment. These are known as "replicated summary tables". For more overview information on such tables, refer to *Administration Guide: Planning*.

Note: Summary tables are not used with static SQL or nicknames.

In general a summary table, or a replicated summary table, is used for optimization of a query if the isolation level of the summary table, or the replicated summary table, is higher than or equal to the isolation level of the query. For example, if a query is running under the cursor stability (CS) isolation level, only summary tables, and replicated summary tables, that are defined under CS or higher isolation levels are used for optimization.

To create a summary table, you use the CREATE SUMMARY TABLE statement with the AS *fullselect* clause and the IMMEDIATE or REFRESH DEFERRED options.

You have the option of uniquely identifying the names of the columns of the summary table. The list of column names must contain as many names as there are columns in the result table of the full select. A list of column names must be given if the result table of the full select has duplicate column names or has an unnamed column. An unnamed column is derived from a constant, function, expression, or set operation that is not named using the AS clause of the select list. If a list of column names is not specified, the columns of the table inherit the names of the columns of the result set of the full select.

When you create the summary table, you have the option of specifying whether the summary table is refreshed automatically when the base table is

changed, or whether it is refreshed by using the REFRESH TABLE statement. To have the summary table refreshed automatically when changes are made to the base table or tables, specify the REFRESH IMMEDIATE keyword. An immediate refresh is useful when:

- You have queries that take a long time to complete when run against a base table
- The base table or tables are infrequently changed
- The refresh is not expensive.

The summary table, in this situation, can provide pre-computed results. If you want the refresh of the summary table to be deferred, specify the REFRESH DEFERRED keyword. Summary tables specified with REFRESH DEFERRED will **not** reflect changes to the underlying base tables. You should use summary tables where this is not a requirement. For example, if you run DSS queries, you would use the summary table to contain legacy data.

A summary table defined with REFRESH DEFERRED may be used in place of a query when it:

- Conforms to the restrictions for a fullselect of a refresh immediate summary table, except:
  - The SELECT list is not required to include COUNT(*) or COUNT_BIG(*)
  - The SELECT list can include MAX and MIN column functions
  - A HAVING clause is allowed.

The SQL special register CURRENT REFRESH AGE SQL is set to ANY or has a value of 99999999999999. The collection of nines is the maximum value allowed in this special register which is a timestamp duration value with a data type of DECIMAL(20,6).

Note: Summary tables defined with REFRESH DEFERRED are not used to optimize static SQL.

You use the CURRENT REFRESH AGE special register to specify the amount of time that the summary table with deferred refresh can be used for a dynamic query before it must be refreshed. To set the value of the CURRENT REFRESH AGE special register, you can use the SET CURRENT REFRESH AGE statement. For more information about the CURRENT REFRESH AGE special register and the SET CURRENT REFRESH AGE statement, refer to the *SQL Reference*.

Summary tables defined with REFRESH IMMEDIATE are applicable to both static and dynamic queries and do not need to use the CURRENT REFRESH AGE special register.

Note: Setting the CURRENT REFRESH AGE special register to a value other than zero should be done with caution. By allowing a summary table that may not represent the values of the underlying base table to be used to optimize the processing of the query, the result of the query may *not* accurately represent the data in the underlying table. This may be reasonable when you know the underlying data has not changed, or you are willing to accept the degree of error in the results based on your knowledge of the data.

With activity affecting the source data, a summary table over time will no longer contain accurate data. You will need to use the REFRESH TABLE statement. Refer to the *SQL Reference* for more information.

If you want to create a new base table that is based on any valid *fullselect*, specify the DEFINITION ONLY keyword when you create the table. When the create table operation completes, the new table is not treated as a summary table, but rather as a base table. For example, you can create the exception tables used in LOAD and SET INTEGRITY as follows:

```
CREATE TABLE XT AS
   (SELECT T.*, CURRENT TIMESTAMP AS TIMESTAMP,CLOB(",32K)
   AS MSG FROM T) DEFINITION ONLY
```

Here are some of the key restrictions regarding summary tables:

1. You cannot alter a summary table.
2. You cannot alter the length of a column for a base table if that table has a summary table.
3. You cannot import data into a summary table.
4. You cannot create a unique index on a summary table.
5. You cannot create a summary table based on the result of a query that references one or more nicknames.

Refer to the *SQL Reference* for a complete statement of summary table restrictions.

Creating an Alias

An alias is an indirect method of referencing a table, nickname, or 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 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 statement, except for table check-constraint definitions, in which an existing table or view name can be referenced.

An alias name can be used wherever an existing table name can be used, and can refer to another alias if no circular or repetitive references are made along the chain of aliases.

The alias name cannot be the same as an existing table, view, or alias, and can only refer to a table within the same database. The name of a table or view used in a CREATE TABLE or CREATE VIEW statement cannot be the same as an alias name in the same schema.

You do not require special authority to create an alias, unless the alias is in a schema other than the one owned by your current authorization ID, in which case DBADM authority is required.

An alias can be defined for a table, view, or alias that does not exist at the time of definition. However, it 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.

To create an alias using the Control Center:

1. Expand the object tree until you see the **Aliases** folder.
2. Right-click the **Aliases** folder, and select **Create** from the pop-up menu.
3. Complete the information, and click **Ok**.

To create an alias using the command line, enter:

```
CREATE ALIAS <alias_name> FOR <table_name>
```

The alias is replaced at statement compilation time by the table or view name. If the alias or alias chain cannot be resolved to a table or view name, an error results. For example, if WORKERS is an alias for EMPLOYEE, then at compilation time:

```
SELECT * FROM WORKERS
```

becomes in effect

```
SELECT * FROM EMPLOYEE
```

The following SQL statement creates an alias WORKERS for the EMPLOYEE table:

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

Note: DB2 for MVS/ESA employs two distinct concepts of aliases: ALIAS and SYNONYM. These two concepts differ from DB2 Universal Database as follows:

- ALIASes in DB2 for MVS/ESA:
  - Require their creator to have special authority or privilege
  - Cannot reference other aliases.
- SYNONYMs in DB2 for MVS/ESA:
  - Can only be used by their creator
  - Are always unqualified
  - Are dropped when a referenced table is dropped
  - Do not share namespace with tables or views.

Creating a Wrapper

In a federated database, the CREATE WRAPPER statement registers a wrapper. The statement defines the mechanism by which a federated server can interact with a certain category of data source.

Specific libraries must be used for specific data source types, versions, communication protocols, and operating systems. For example, AS/400 and DB2 for OS/390 data sources are accessed using the `libdrda.dll` library for federated databases operating on Windows NT operating systems using APPC communications.

You must have SYSADM or DBADM authority at the federated database to use the CREATE WRAPPER statement.

Creating a wrapper from the Control Center or from the command line processor registers it to the federated database.

To create a wrapper using the Control Center:

1. Expand the object tree until you see the **Federated Database Objects** folder.
2. Right-click the **Federated Database Objects** folder, and select **Create wrapper** from the pop-up menu.
3. Complete the information, and click **Ok**.

To create a wrapper using the command line, enter:

```
CREATE WRAPPER <wrapper_name> LIBRARY '<library_name>'
```

The following SQL statement registers the wrapper ORACLE8 on a Windows NT operating system:

```
CREATE WRAPPER ORACLE8 LIBRARY 'libnet8.dll'
```

For details on using the CREATE WRAPPER statement, refer to the *SQL Reference*.

Creating a Server

In a federated database, create servers to define data sources to DB2 and describe their characteristics: name, wrapper, type, version, location, and options. This information is used to map nicknames to specific data management systems and to provide information to the DB2 optimizer. Server information is located in the SYSCAT.SERVERS and SYSCAT.SERVEROPTIONS catalog views.

Note: In this section, servers represent data sources, not DRDA servers or DB2 servers. To access other data sources (for example, Oracle), DB2 Connect is required.

You can create a server object only if a wrapper has been created.

You must have SYSADM or DBADM authority at the federated database to use this statement.

You can create user mappings to manage differences in authentication processing between DB2 and data source servers. User mappings are discussed in detail in "User Mappings" on page 238.

When a server is dropped, all objects dependent on that server are dropped (such as user mappings, nicknames, function mappings, type mappings, and plans).

Provide server options when creating a server. These options contain necessary details about the server (such as the node name). Server options can also set specific performance and security values.

You can create servers from the Control Center or the command line processor.

To create a server using the Control Center:

1. Expand the object tree until you see the **Servers** folder under the **Federated Database Objects** folder.
2. Right-click the **Servers** folder, and select **Create server** from the pop-up menu.
3. Complete the information, and click **Ok**.

To create a server using the command line, enter:

```
CREATE SERVER <server_name> TYPE <server_type>
   VERSION <server_version> WRAPPER <wrapper_name>
   OPTIONS (<server_option_name> <string_constant>)
```

The following sample SQL statement creates the Oracle server ORA8:

```
CREATE SERVER ORA8 TYPE ORACLE VERSION 8 WRAPPER ORACLE8 OPTIONS
(NODE 'ONODE')
```

The following sample SQL statement creates the DB2 server DB2TEST:

```
CREATE SERVER DB2TEST TYPE DB2 VERSION 6.1 WRAPPER DB2UDB OPTIONS
(NODE 'DB2TEST', DBNAME 'TEST1')
```

The definition of NODE, in SERVER SQL statements, varies depending on the data source. If the data source is a DB2 DBMS, the value refers to an instance of DB2 that has one or more databases. In the previous example, note that the DBNAME option specifies the database name. If the data source is a DB2 for OS/390 DBMS, the value refers to the LOCATION name of the DB2 for OS/390 system. If the data source is an Oracle DBMS, the DBNAME option is not needed because an Oracle instance contains only one database.

For additional details about the CREATE SERVER statement syntax, refer to the *SQL Reference*. For additional details about using the CREATE SERVER statement, refer to the *Installation and Configuration Supplement*.

Using Server Options to Help Define Data Sources and Facilitate Authentication Processing

You can set variables called *server options* to values that affect how a federated server accesses data sources. This section:

- Explains the purpose of server options
- Describes what SQL statements you use to specify server options
- Shows the server options and their settings

Purposes of Server Options

In general, you use server options to:

- Supply and update information about data sources. A server reference includes both basic information about a data source—for example, its name—and information that can change over time. Some of the changeable information is conveyed by values assigned to server options. For example, the value assigned to the cpu_ratio option indicates whether the data source's CPU is faster or slower than the DB2 system CPU. If the DB2 system gets one or more processor upgrades, this value should change.
- Facilitate authentication. You can set some server options to ensure that user IDs and passwords are sent to the data source in the proper case. For example, you can set the fold_id option so that before the federated server

sends a user ID to a data source, the federated server transforms the name to the case (upper or lower) that the data source requires. Alternatively, if you define the user ID to the federated server in the required case, you can set the fold_id option to prevent the server from trying to change the case and consuming overhead in the process.

- Optimize queries. Some server options and their values facilitate optimization. For example: in the CREATE SERVER statement, you can specify certain performance statistics as option values. Specifically, you can set the cpu_ratio option to a value that indicates the relative speeds of the data source's and federated server's CPUs. And you can set the io_ratio option to a value that indicates the relative rates of the I/O devices of the data source and federated server. When you run CREATE SERVER, these statistics are added to the catalog view SYSCAT.SERVEROPTIONS, and the optimizer uses them in developing its access plan for the data source. If a statistic changes (as might happen, for instance, if the data source CPU is upgraded), you can use the ALTER SERVER statement to update SYSCAT.SERVEROPTIONS with this change. The optimizer then uses your update in developing its next access plan for the data source.

SQL for Server Options

There are three SQL statements in which you can assign values to server options: CREATE SERVER, ALTER SERVER, and SET SERVER OPTION.

Use the CREATE SERVER statement to set an option to a value that persists indefinitely over time for multiple connections to a data source. With this statement, you can set an option to a value other than the default or, if an option has no default value, you can set it to an initial value.

Use the ALTER SERVER statement if, after setting a server option to a value with the CREATE SERVER statement, you want to set it to a different value that persists over multiple connections.

Use the SET SERVER OPTION statement to change server option values temporarily for the duration of a single connection to a database. SET SERVER OPTION statements must be issued first within the first unit of work following the connection to the data source.

For example, to temporarily enable the use of plan hints for the Oracle server ORASEB1, issue the statement:

```
SET SERVER OPTION plan_hints TO 'Y' FOR SERVER ORASEB1
```

Server Options and Their Settings

The table below describes the server options and the values that you can set them to. Unless otherwise stated, all server option values must be enclosed in single quotes.

Table 2. Server Options and Their Settings

Option	Valid Settings	Default Setting
collating_sequence	Specifies whether the data source uses the same default collating sequence as the federated database, based on the code set and the country information. If a data source has a collating sequence that differs from DB2's collating sequence, most operations depending on DB2's collating sequence cannot be remotely evaluated at a data source. An example is executing MAX column functions against a nickname character column at a data source with a different collating sequence. Because results might differ if the MAX function is evaluated at the remote data source, DB2 will perform the aggregate operation and the MAX function locally. If your query contains an equal sign, it is possible to push-down that portion of the query even if the collating sequences are different (set to 'N'). For example, the predicate C1 = 'A' could be pushed-down to a data source. Of course, such queries cannot be pushed-down when the collating sequence at the data source is case-insensitive. When a data source is case-insensitive, the results from C1= 'A' and C1 = 'a' are the same, which is not acceptable in a case-sensitive environment (DB2). Administrators can create federated databases with a particular collating sequence that matches the data source collating sequence. This approach may speed performance if all data sources use the same collating sequence or if most or all column functions are directed against data sources that use the same collating sequence. 'Y' Data source's collating sequence is the same as federated database's. 'N' Data source's collating sequence is not the same as federated database's. 'I' Data source's collating sequence is different from federated database's and is case-insensitive (for example, 'TOLLESON' and 'TolLESon' are considered equal).	'N'
comm_rate	Specifies the communication rate between a federated server and its associated data sources. Expressed in megabytes per second.	'2.0'

Table 2. Server Options and Their Settings (continued)

Option	Valid Settings	Default Setting
connectstring	Specifies initialization properties needed to connect to an OLE DB provider. For the complete syntax and semantics of the connection string, see the "Data Link API of the OLE DB Core Components" in the *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, Microsoft Press, 1998*.	None
cpu_ratio	Indicates how much faster or slower a data source's CPU runs than the federated server's CPU. A value of "1.0" indicates an equal speed between the data source's CPU and the federated server's CPU. A value <1.0 indicates a slower data source CPU. A value >1.0 indicates a faster data source CPU.	'1.0'
dbname	Name of the data source database that you want the federated server to access. Required for DB2 family data sources; does not apply to Oracle** data sources.	None.
fold_id (See notes 1 and 4 at the end of this table.)	Applies to user IDs that the federated server sends to data sources for authentication. Valid values are: 'U' The federated server folds the user ID to uppercase before sending it to the data source. This is a logical choice for DB2 Family and Oracle** data sources (See note 2 at end of this table.) 'N' The federated server does nothing to the user ID before sending it to the data source. (See note 2 at end of this table.) 'L' The federated server folds the user ID to lowercase before sending it to the data source. If none of these settings are used, the federated server tries to send the user ID to the data source in uppercase. If the user ID fails, the server tries sending it in lowercase.	None.
fold_pw (See notes 1, 3 and 4 at the end of this table.)	Applies to passwords that the federated server sends to data sources for authentication. Valid values are: 'U' The federated server folds the password to uppercase before sending it to the data source. This is a logical choice for DB2 Family and Oracle** data sources. 'N' The federated server does nothing to the password before sending it to the data source. 'L' The federated server folds the password to lowercase before sending it to the data source. If none of these settings are used, the federated server tries to send the password to the data source in uppercase. If the password fails, the server tries sending it in lowercase.	None.

Table 2. Server Options and Their Settings (continued)

Option	Valid Settings	Default Setting
io_ratio	Denotes how much faster or slower a data source's I/O system runs than the federated server's I/O system. A value of "1.0" indicates an equal speed between the data source's CPU and the federated server's CPU. A value <1.0 indicates a slower data source CPU. A value >1.0 indicates a faster data source CPU.	'1.0'
node	Name by which a data source is defined as an instance to its RDBMS. Required for all data sources. For a DB2 family data source, this name is the node specified in the federated database's `DB2 node` directory. To view this directory, issue the **db2 list node directory** command. For an Oracle** data source, this name is the server name specified in the Oracle** `tnsnames.ora` file. To access this name on the Windows NT platform, specify the **View Configuration Information** option of the Oracle** SQL Net Easy Configuration tool.	None.
password	Specifies whether passwords are sent to a data source. 'Y' Passwords are always sent to the data source and validated. This is the default value. 'N' Passwords are not sent to the data source (regardless of any user mappings) and not validated. 'ENCRYPTION' Passwords are always sent to the data source in encrypted form and validated. Valid only for DB2 Family data sources that support encrypted passwords.	'Y'
plan_hints	Specifies whether *plan hints* are to be enabled. Plan hints are statement fragments that provide extra information for data source optimizers. This information can, for certain query types, improve query performance. The plan hints can help the data source optimizer decide whether to use an index, which index to use, or which table join sequence to use. 'Y' Plan hints are to be enabled at the data source if the data source supports plan hints. 'N' Plan hints are not to be enabled at the data source.	'N'

Table 2. Server Options and Their Settings (continued)

Option	Valid Settings		Default Setting
pushdown	'Y'	DB2 will consider letting the data source evaluate operations.	'Y'
	'N'	DB2 will retrieve only columns from the remote data source and will not let the data source evaluate other operations, such as joins.	
varchar_no_trailing_blanks	Specifies if this data source uses non-blank padded VARCHAR comparison semantics. For varying-length character strings that contain no trailing blanks, some DBMS' s non-blank-padded comparison semantics return the same results as DB2's comparison semantics. If you are certain that all VARCHAR table/view columns at a data source contain no trailing blanks, consider setting this server option to 'Y' for a data source. This option is often used with Oracle** data sources. Ensure that you consider all objects that can potentially have nicknames (including views).		'N'
	'Y'	This data source has non-blank-padded comparison semantics similar to DB2's.	
	'N'	This data source does not have the same non-blank-padded comparison semantics as DB2's.	

Notes on this table:

1. This field is applied regardless of the value specified for authentication.
2. Because DB2 stores user IDs in uppercase, the values 'N' and 'U' are logically equivalent to each other.
3. The setting for fold_pw has no effect when the setting for password is 'N'. Because no password is sent, case cannot be a factor.
4. Avoid null settings for either of these options. A null setting may seem attractive because DB2 will make multiple attempts to resolve user IDs and passwords; however, performance might suffer (it is possible that DB2 will send a user ID and password four times before successfully passing data source authentication).

Using Pass-Through Sessions with Servers

Pass-through sessions let applications communicate directly with a server using the server's native client access method and native SQL dialect.

Pass-through sessions are useful when:

- Applications must create objects at the data source or perform INSERT, UPDATE, or DELETE operations

- DB2 does not support a unique data source operation

When referencing objects in a pass-through session, use the true name of the object (not the nickname).

Use the SET PASSTHRU statement to start a pass-through session and access a server directly. This statement must be issued dynamically. An example of this statement is:

```
SET PASSTHRU BACKEND
```

which opens a pass-through session to the data source BACKEND.

For more information on SET PASSTHRU and SQL processing in pass-through sessions, see the *SQL Reference*.

Creating a Nickname

In a federated database, nicknames are identifiers for data source tables, aliases, and views. Distributed requests typically reference nicknames, not data source tables or views.

Nicknames are part of the means by which DB2 provides location transparency. Nicknames rely on server definitions for data source location information to find and efficiently access data sources. An ALTER SERVER statement can, for example, transparently update server performance data and version information for all users and applications without requiring new nicknames or changes to application code.

Nicknames can be created in the Control Center or from the command line processor. You can define more than one nickname for the same data source table or view.

Nicknames cannot be used in static SQL statements.

Before creating a nickname, run the equivalent of the RUNSTATS command at the data source and update statistics for data source objects. Statistical information is gathered from data sources when a nickname is created and stored in the federated database catalog. This catalog data includes table and column definitions, and, if available, index definitions and statistics.

The following SQL statement creates the nickname CUSTOMER:

```
CREATE NICKNAME CUSTOMER for OS390A.SHAWNB.CUSTLIST
```

You must hold one of the SYSADM or DBADM authorities, or, you must have either the database privilege IMPLICIT_SCHEMA or the schema privilege CREATEIN (for the current schema) at the federated database to use this statement.

For additional details on using the CREATE NICKNAME statement, refer to the *SQL Reference*.

Referencing Nickname and Data Source Objects

References to data source objects typically use the defined nickname. The one exception is a reference within a pass-through session (see "Using Pass-Through Sessions with Servers" on page 158 for more information). For example, if you define the nickname DEPT for the data source table DB2MVS1.PERSON.DEPT, the statement SELECT * FROM DEPT is allowed; the statement SELECT * FROM DB2MVS1.PERSON.DEPT is not allowed.

Working with Nickname and Data Source Objects

Most utility commands (LOAD, IMPORT, EXPORT, REORGCHK, REORGANIZE TABLE) do not support nicknames.

COMMENT ON is supported; it updates the system catalog at the federated database.

Insert, update, and delete operations are not supported against nicknames.

Identifying Existing Nicknames and Data Sources

After you have created several nicknames, you might want to use the following information to identify to which data source a given nickname corresponds or identify all nicknames at a given data source.

Identifying a Nickname and Its Data Source

This example assumes that you know the nickname (*PAYROLL*) and who created it (*ACCTG*), but need additional information about the data source. Use the following SQL statement to first obtain information about what *PAYROLL* is known as at its data source (SERVER).

```
select option, setting
  from syscat.taboptions
  where tabname = 'PAYROLL'
     and tabschema = 'ACCTG'
     and option in ('SERVER','REMOTE_SCHEMA','REMOTE_TABLE');
```

The answer set from this statement is `DB2_MVS, FINANCE, DEPTJ35_PAYROLL`. You now know that *PAYROLL* is the nickname for the table called DEPTJ35_PAYROLL owned by FINANCE at the server named DB2_MVS. You can use this information in a subsequent SELECT statement:

```
select option,setting
    from syscat.serveroptions
    where servername = 'DB2_MVS'
       and option in ('NODE','DBNAME');
```

The answer set from this statement is REGIONW and DB2MVSDB3. You now know that the table DEPTJ35_PAYROLL is in a database named DB2MVSDB3, on a node called REGIONW.

With this information, you can use the LIST NODE DIRECTORY command to obtain information about the REGIONW node, such as the communications protocol and security type used. If the node had been for a data source other than the DB2 Family, you would need to check that data source's configuration files to find similar information. For example, if the node had been an Oracle data source, you would get similar information from the Oracle tnsnames.ora file.

For details on system catalog views, refer to the *SQL Reference*.

Identifying All Nicknames Known to DB2

The following SQL statement provides a list of all nicknames known to the federated database, including the schema name and remote server for each nickname.

```
select tabname,tabschema, setting as remote_server
   from syscat.taboptions
   where option = 'SERVER';
```

Creating an Index, Index Extension, or an Index Specification

An index is a list of the locations of rows, sorted by the contents of one or more specified columns. Indexes are typically used to speed up access to a table. However, they can also serve a logical data design purpose. For example, a *unique index* does not allow entry of duplicate values in the columns, thereby guaranteeing that no two rows of a table are the same. Indexes can also be created to specify ascending or descending order of the values in a column.

An index extension is an index object for use with indexes that have structured type or distinct type columns.

An index specification is a metadata construct. It tells the optimizer that an index exists for a data source object (table or view) referenced by a nickname. An index specification does not contain lists of row locations–it is just a description of an index. The optimizer uses the index specification to improve access to the object indicated by the nickname. When a nickname is first created, an index specification is generated if an index exists for the underlying table at the data source in a format DB2 can recognize.

Note: If needed, create index specifications on table nicknames or view nicknames where the view is over one table.

Manually create an index or an index specification when:

- It would improve performance. For example, if you want to encourage the optimizer to use a particular table or nickname as the inner table of a nested loop join, create an index specification on the joining column if no index exists. Refer to the *Administration Guide: Performance* for more information about when you would want an index or an index specification.
- An index for a base table was added after the nickname for that table was created.

Index specifications can be created when no index exists on the base table (DB2 will not check for the remote index when you issue the CREATE INDEX statement). An index specification does not enforce uniqueness of rows even when the UNIQUE keyword is specified.

The DB2 Index Advisor is a wizard that assists you in choosing an optimal set of indexes. You can access this wizard through the Control Center. The comparable utility is called *db2advis*.

An index is defined by columns in the base table. It can be defined by the creator of a table, or by a user who knows that certain columns require direct access. A primary index key is automatically created on the primary key, unless a user-defined index already exists.

Any number of indexes can be defined on a particular base table, and they can have a beneficial effect on the performance of queries. However, the more indexes there are, the more the database manager must modify during update, delete, and insert operations. Creating a large number of indexes for a table that receives many updates can slow down processing of requests. Therefore, use indexes only where a clear advantage for frequent access exists.

Any column that is part of an index key is limited to 255 bytes.

Note: The DB2_INDEX_2BYTEVARLEN registry variable can be used to allow columns with a length greater than 255 bytes to be specified as part of an index key.

The maximum number of columns in an index is 16. If you are indexing a typed table, the maximum number of columns is 15. The maximum length of an index key is 1024 bytes. As previously mentioned, many index keys on a table can slow down processing of requests. Similarly, large index keys can also slow down processing requests.

An *index key* is a column or collection of columns on which an index is defined, and determines the usefulness of an index. Although the order of the columns making up an index key does not make a difference to index key creation, it may make a difference to the optimizer when it is deciding whether or not to use an index.

If the table being indexed is empty, an index is still created, but no index entries are made until the table is loaded or rows are inserted. If the table is not empty, the database manager makes the index entries while processing the CREATE INDEX statement.

For a *clustering index*, new rows are inserted physically close to existing rows with similar key values. This yields a performance benefit during queries because it results in a more linear access pattern to data pages and more effective pre-fetching.

If you want a primary key index to be a clustering index, a primary key should not be specified at CREATE TABLE. Once a primary key is created, the associated index cannot be modified. Instead, perform a CREATE TABLE without a primary key clause. Then issue a CREATE INDEX statement, specifying clustering attributes. Finally, use the ALTER TABLE statement to add a primary key that corresponds to the index just created. This index will be used as the primary key index.

Generally, clustering is more effectively maintained if the clustering index is unique.

Column data which is not part of the unique index key but which is to be stored/maintained in the index is called an *include* column. Include columns can be specified for unique indexes only. When creating an index with include columns, only the unique key columns are sorted and considered for uniqueness. Use of include columns improves the performance of data retrieval when index access is involved.

The database manager uses a B+ tree structure for storing indexes where the bottom level consists of `leaf nodes`. The leaf nodes or pages are where the actual index key values are stored. When creating an index, you can enable those index leaf pages to be merged or reorganized online. Online index reorganization is used to prevent the situation where, after much delete and update activity, many leaf pages of an index have only a few index keys left on them. In such a situation, and without online reorganization, space could only be reclaimed by an off-line reorganization of the data and index. When deciding whether to create an index with the ability to reorganize index pages online, you should consider this question: Is the added performance cost of checking for space to merge each time a key deletion occurs and the actual cost to complete the merge, if there is enough space, greater than the benefit

of better space utilization for the index and less than a reduced need to perform an off-line reorganization to reclaim space?

Note: Pages freed after an online reorganization merge are available for re-use only for other indexes in the same table. With a full reorganization, those pages that are freed are available to other objects (when working with Database Managed Storage) or to disk space (when working with System Managed Storage). In addition, an online reorganization will not free up any non-leaf pages of the index, whereas a full reorganization will make the index as small as possible by making the index as small as possible, reducing the non-leaf and leaf pages as well as the number of levels of the index.

See "Using the CREATE INDEX Statement" on page 165 for more information on how to implement an index that will reorganize online.

Indexes for tables in a partitioned database are built using the same CREATE INDEX statement. They are partitioned based on the partitioning key of the table. An index on a table consists of the local indexes in that table on each node in the nodegroup. Note that unique indexes defined in a multiple partition environment must be a superset of the partitioning key.

Performance Tip: If you are going to carry out the following series of tasks:

1. Create Table
2. Load Table
3. Create Index
4. Perform RUNSTATS

then you should consider ordering the execution of tasks in the following way:

1. Create the table
2. Create the index
3. Load the table with the `statistics yes` option requested.

Refer to *Data Movement Utilities Guide and Reference* for more information on LOAD performance improvements.

Indexes are maintained after they are created. Subsequently, when application programs use a key value to randomly access and process rows in a table, the index based on that key value can be used to access rows directly. This is important, because the physical storage of rows in a base table is not ordered. When a row is inserted, unless there is a clustering index defined, the row is placed in the most convenient storage location that can accommodate it. When searching for rows of a table that meet a particular selection condition and the

table has no indexes, the entire table is scanned. An index optimizes data retrieval without performing a lengthy sequential search.

The data for your indexes can be stored in the same table space as your table data, or in a separate table space containing index data. The table space used to store the index data is determined when the table is created (see "Creating a Table in Multiple Table Spaces" on page 134).

To create an index using the Control Center:

1. Expand the object tree until you see the **Indexes** folder.
2. Right-click the **Indexes** folder, and select **Create —> Index Using Wizard** from the pop-up menu.
3. Follow the steps in the wizard to complete your task.

To create an index using the command line, enter:

```
CREATE INDEX <name> ON <table_name> (<column_name>)
```

The following two sections, "Using an Index" and "Using the CREATE INDEX Statement", provide more information on creating an index.

Using an Index

An index is never directly used by an application program. The decision on whether to use an index and which of the potentially available indexes to use is the responsibility of the optimizer.

The best index on a table is one that:

- Uses high-speed disks
- Is highly-clustered
- Is made up of only a few narrow columns

For a detailed discussion of how an index can be beneficial, refer to "Index Scan Concepts" in the *Administration Guide: Performance*.

Using the CREATE INDEX Statement

You can create an index that will allow duplicates (a non-unique index) to enable efficient retrieval by columns other than the primary key, and allow duplicate values to exist in the indexed column or columns.

The following SQL statement creates a non-unique index called LNAME from the LASTNAME column on the EMPLOYEE table, sorted in ascending order:

```
CREATE INDEX LNAME ON EMPLOYEE (LASTNAME ASC)
```

The following SQL statement creates a unique index on the phone number column:

```
CREATE UNIQUE INDEX PH ON EMPLOYEE (PHONENO DESC)
```

A unique index ensures that no duplicate values exist in the indexed column or columns. The constraint is enforced at the end of the SQL statement that updates rows or inserts new rows. This type of index cannot be created if the set of one or more columns already has duplicate values.

The keyword ASC puts the index entries in ascending order by column, while DESC puts them in descending order by column. The default is ascending order.

You can create a unique index on two columns, one of which is an include column. The primary key is defined on the column that is not the include column. Both of them are shown in the catalog as primary keys on the same table. Normally there is only one primary key per table.

The INCLUDE clause specifies additional columns to be appended to the set of index key columns. Any columns included with this clause are not used to enforce uniqueness. The included columns may improve the performance of some queries through index-only access. The columns must be distinct from the columns used to enforce uniqueness (otherwise you will receive error message SQLSTATE 42711). The limits for the number of columns and sum of the length attributes apply to all of the columns in the unique key and in the index.

A check is performed to determine if an existing index matches the primary key definition (ignoring any INCLUDE columns in the index). An index definition matches if it identifies the same set of columns without regard to the order of the columns or the direction (either ascending or descending) specifications. If a matching index definition is found, the description of the index is changed to indicate that it is the primary index, as required by the system, and it is changed to unique (after ensuring uniqueness) if it was a non-unique index.

This is why it is possible to have more than one primary key on the same table as indicated in the catalog.

When working with a structured type, it might be necessary to create user-defined index types. This requires a means of defining index maintenance, index search, and index exploitation functions. Refer to the *SQL Reference* for information on the requirements for creating an index type.

The following SQL statement creates a clustering index called INDEX1 on the LASTNAME column of the EMPLOYEE table:

```
CREATE INDEX INDEX1 ON EMPLOYEE (LASTNAME) CLUSTER
```

To use the internal storage of the database effectively, use clustering indexes with the PCTFREE parameter associated with the ALTER TABLE statement so that new data can be inserted on the correct pages. When data is inserted on the correct pages, clustering order is maintained. Typically, the greater the INSERT activity on the table, the larger the PCTFREE value (on the table) that will be needed in order to maintain clustering. Since this index determines the order by which the data is laid out on physical pages, only one clustering index can be defined for any particular table.

If, on the other hand, the index key values of these new rows are, for example, always new high key values, then the clustering attribute of the table will try to place them at the end of the table. Having free space in other pages will do little to preserve clustering. In this case, placing the table in append mode may be a better choice than a clustering index and altering the table to have a large PCTFREE value. You can place the table in append mode by issuing: ALTER TABLE APPEND ON. See "Changing Table Attributes" on page 201 for additional overview information on ALTER TABLE. Refer to the *SQL Reference* for additional detailed information on ALTER TABLE.

The above discussion also applies to new "overflow" rows that result from UPDATEs that increase the size of a row.

The MINPCTUSED clause of the CREATE INDEX statement specifies the threshold for the minimum amount of used space on an index leaf page. If this clause is used, online index reorganization is enabled for this index. Once enabled, the following considerations are used to determine if an online reorganization takes place: After a key is deleted from a leaf page of this index and a percentage of used space on the page is less than the specified threshold value, the neighboring index leaf pages are checked to determine if the keys on the two leaf pages can be merged into a single index leaf page.

For example, the following SQL statement creates an index with online index reorganization enabled:

```
CREATE INDEX LASTN ON EMPLOYEE (LASTNAME) MINPCTUSED=20
```

When a key is deleted from this index, if the remaining keys on the index page take up twenty percent or less space on the index page, then an attempt is made to delete an index page by merging the keys of this index page with those of a neighboring index page. If the combined keys can all fit on a single page, this merge is performed and one of the index pages is deleted.

The PCTFREE clause of the CREATE INDEX statement specifies the percentage of each index page to leave as free space when the index is built. Leaving more free space on the index pages will result in fewer page splits.

This will reduce the need to reorganize the table in order to regain sequential index pages which increases prefetching. And prefetching is one important component that may improve performance. Again, if there are always high key values, then you will want to consider lowering the value of the PCTFREE clause of the CREATE INDEX statement. In this way there will be limited wasted space reserved on each index page.

If you have a replicated summary table, its base table (or tables) must have a unique index, and the index key columns must be used in the query that defines the replicated summary table. Refer to *Administration Guide: Planning* for more information on replicated summary tables.

For intra-partition parallelism, create index performance is improved by using multiple processors for the scanning and sorting of data that is performed during index creation. The use of multiple processors is enabled by setting *intra_parallel* to YES(1) or ANY(-1). The number of processors used during index creation is determined by the system and is not affected by the configuration parameters *dft_degree* or *max_querydegree*, by the application runtime degree, or by the SQL statement compilation degree. If the database configuration parameter *indexsort* is NO, then index creation will not use multiple processors.

In multiple partition databases, unique indexes must be defined as supersets of the partitioning key.

Creating a User-Defined Extended Index Type

To support user-defined index types, DB2 Universal Database allows you to create and apply your own logic for the primary components that make up how an index works. Those components that can be substituted are:

- Index maintenance. This allows the ability to map index column content to an index key. Such a mapping is done through a user-defined mapping function. Exactly one structured type column can participate in an extended index. Unlike an ordinary index, an extended index may have more than one index entry per row. Multiple index entries per row could enable a text document to be stored as an object with a separate index entry for each keyword in the document.
- Index exploitation. This enables the application designer to associate filtering conditions (range predicates) with a user-defined function (UDF) that would otherwise be opaque to the optimizer. This enables DB2 to avoid making a separate UDF call for each row, and thereby avoids context switching between client and server, greatly improving performance.

Note: The user-defined function definition must be deterministic and must not allow external actions in order to be exploitable by the optimizer.

An optional data filter function can also be specified. The optimizer uses the filter against the fetched tuple before the user-defined function is evaluated.

Only a structured type or distinct type column can use the index extension to create a user-defined extended index type on these objects. The user-defined extended index type must not:

- Be defined with clustering indexes
- Have INCLUDE columns

Details on Index Maintenance

You define two of the components that make up the operations of an index through the CREATE INDEX EXTENSION statement.

Index maintenance is the process of transforming the index column content (or source key) to a target index key. The transformation process is defined using a table function that has previously been defined in the database.

The FROM SOURCE KEY clause specifies a structured data type or distinct type for the source key column supported by this index extension. A single parameter name and data type are given and associated with the source key column.

The GENERATE KEY USING clause specifies the user-defined table function used to generate the index key. The output from this function must be specified in the TARGET KEY clause specification. The output from this function can also be used as input for the index filtering function specified on the FILTER USING clause.

Details on Index Searching

Index searching maps search arguments to search ranges.

The WITH TARGET KEY clause of the CREATE INDEX EXTENSION statement specifies the target key parameters that are the output of the user-defined table function specified on the GENERATE KEY USING clause. A single parameter name and data type are given and associated with the target key column. This parameter corresponds to the columns of the RETURNS table of the user-defined table function of the GENERATE KEY USING clause.

The SEARCH METHODS clause introduces one or more search methods defined for the index. Each search method consists of a method name, search arguments, a range producing function, and an optional index filter function. Each search method defines how index search ranges for the underlying user-defined index are produced by a user-defined table function. Further,

each search method defines how the index entries in a particular search range can be further qualified by a user-defined scalar function to return a single value.

- The WHEN clause associates a label with a search method. The label is an SQL identifier that relates to the method name specified in the index exploitation rule (found in the PREDICATES clause of a user-defined function). One or more parameter names and data types are given for use as arguments in the range function and/or index filtering function. The WHEN clause specifies the action that can be taken by the optimizer when the PREDICATES clause of the CREATE FUNCTION statement matches an incoming query.
- The RANGE THROUGH clause specifies the user-defined external table function that produces index key ranges. This enables the optimizer to avoid calling the associated UDF when the index keys fall outside the key ranges.
- The FILTER USING clause is an optional way of specifying a user-defined external table function or a case expression used to filter index entries returned from the range-producing function. If the value returned by the index filter function or case expression is 1, the row corresponding to the index entry is retrieved from the table. If the value returned is something other than 1, the index entry is discarded. This feature is valuable when the cost of the secondary filter is low compared to the cost of evaluating the original method, and the selectivity of the secondary filter is relatively low.

Details on Index Exploitation

Index exploitation occurs in the evaluation of the search method.

The CREATE FUNCTION (External Scalar) statement creates a user-defined predicate used with the search methods defined for the index extension.

The PREDICATES clause identifies those predicates using this function that can possibly exploit the index extensions (and that can possibly use the optional SELECTIVITY clause for the predicate's search condition). If the PREDICATES clause is specified, the function must be defined as DETERMINISTIC with NO EXTERNAL ACTION.

- The WHEN clause introduces a specific use of the function being defined in a predicate with a comparison operator (=, >, <, and others) and a constant or expression (using the EXPRESSION AS clause). When a predicate uses this function with the same comparison operator and the given constant or expression, filtering and index exploitation may be used. The use of a constant is provided mainly to cover Boolean expressions where the result type is either a 1 or a 0. For all other cases, the EXPRESSION AS clause is the better choice.
- The FILTER USING clause identifies a filter function that can be used to perform additional filtering of the result table. It is an alternative and faster

version of the defined function (used in the predicate) that reduces the number of rows on which the user-defined predicate must be executed to determine if rows qualify. Should the results produced by the index be close to the results expected by the user-defined predicate, then the application of this filter function may be redundant.

- You can optionally define a set of rules for each search method of an index extension to exploit the index. You can also define a search method in the index extension to describe the search targets, the search arguments, and how these can be used to perform the index search.
  - The SEARCH BY INDEX EXTENSION clause identifies the index extension.
  - The optional EXACT clause indicates that the index lookup is exact in its predicate evaluation. This clause tells the database not to apply the original user-provided predicate function or the filter function after the index lookup. If the index lookup is not used, then the original predicate and the filter functions have to be applied. If the EXACT clause is not used, then the original user-provided predicate is applied after the index lookup. The EXACT predicate is useful when the index lookup returns the same results as the predicate. This prevents the query execution from applying the user-defined predicate on the results obtained from the index lookup. If the index is expected to provide only an approximation of the predicate, do not specify the EXACT clause.
  - The WHEN KEY clause defines the search target. Only one search target is specified for a key. The value given following the WHEN KEY clause identifies a parameter name of the function being defined. This clause is evaluated as true when the values of the named parameter are columns that are covered by an index based on the index extension specified.
  - The USE clause defines the search argument. The search argument identifies which method defined in the index extension will be used. The method name given here must match a method defined in the index extension. The one or more parameter values identify parameter names of the function being defined and which must be different from any of the parameter names specified in the search target. The number of parameter values and the data type of each must match the parameters defined for the method in the index extension. The match must be exact for built-in and distinct data types, and be within the same structure types.

A Scenario for Defining an Index Extension

A scenario for defining an index extension follows:

1. Define the structured types (for shapes). Use the CREATE TYPE statement to define a type hierarchy where shape is a supertype and nullshape, point, line, and polygon are subtypes. These structured types model spatial entities. For example, the location of a store is a point; the path of a river is a line; and, the boundary of a business zone is a polygon. A

minimum bounded rectangle (mbr) is an attribute. The gtype attribute identifies whether the associated entity is a point, a line, or a polygon. Geographical boundaries are modeled by numpart, numpoint, and geometry attributes. All other attributes are ignored because they are of no interest to this scenario.

2. Create the index extension.
   - Use the CREATE FUNCTION statement to create functions that are used for key transformation (gridentry), range-producing (gridrange), and index filter (checkduplicate and mbroverlap).
   - Use the CREATE INDEX EXTENSION statement to create the remaining needed components of the index.
3. Create the key transformation which corresponds to the index maintenance component of an index.

```
CREATE INDEX EXTENSION iename (parm_name datatype, ...)
   FROM SOURCE KEY (parm_name datatype)
   GENERATE KEY USING table_function_invocation
   ...
```

   The FROM SOURCE KEY clause identifies the parameter and data type of the key transformation. The GENERATE KEY USING clause identifies the function used to map the source key with the value generated from the function.
4. Define the range-producing and index-filter functions which correspond to the index search component of an index.

```
CREATE INDEX EXTENSION iename (parm_name datatype, ...)
   ...
   WITH TARGET KEY
      WHEN method_name (parm_name datatype, ...)
      RANGE THROUGH range_producing_function_invocation
      FILTER USING index_filtering_function_invocation
```

   The WITH TARGET KEY clause identifies the search method definition. The WHEN clause identifies the method name. The RANGE THROUGH clause identifies the function used to limit the scope of the index to be used. The FILTER USING clause identifies the function used to eliminate unnecessary items from the resulting index values.

   Note: The FILTER USING clause could identify a case expression instead of an index filtering function.
5. Define the predicates to exploit the index extension.

```
CREATE FUNCTION within (x shape, y shape)
   RETURNS INTEGER
   ...
   PREDICATES
      WHEN = 1
```

```
FILTER USING mbrWithin (x..mbr..xmin, ...)
SEARCH BY INDEX EXTENSION grid_extension
WHEN KEY (parm_name) USE method_name(parm_name)
```

The PREDICATES clause introduces one or more predicates that are started with each WHEN clause. The WHEN clause begins the specification for the predicate with a comparison operator followed by either a constant or an EXPRESSION AS clause. The FILTER USING clause identifies a filter function that can be used to perform additional filtering of the result table. This is a cheaper version of the defined function (used in the predicate) that reduces the number of rows on which the user-defined predicate must be executed to determine the rows that qualify. The SEARCH BY INDEX EXTENSION clause specifies where the index exploitation takes place. Index exploitation defines the set of rules using the search method of an index extension that can be used to exploit the index. The WHEN KEY clause specifies the exploitation rule. The exploitation rule describes the search targets and search arguments as well as how they can be used to perform the index search through a search method.

6. Define a filter function.

   ```
   CREATE FUNCTION mbrWithin (...)
   ```

   The function defined here is created for use in the predicate of the index extension.

In order for the query optimizer to successfully exploit indexes created to improve query performance, a SELECTIVITY option is available on function invocation. In cases where you have some idea of the percentage of rows that the predicate may return, you can use the SELECTIVITY option on function invocation to help the DB2 optimizer choose a more efficient access path.

In the following example, the `within` user-defined function computes the center and radius (based on the first and second parameters, respectively), and builds a statement string with an appropriate selectivity:

```
SELECT * FROM customer
   WHERE within(loc, circle(100, 100, 10)) = 1 SELECTIVITY .05
```

In this example, the indicated predicate (`SELECTIVITY .05`) filters out 95 percent of the rows in the `customer` table.

Chapter 4. Altering a Database

This chapter focuses on what you must consider before altering a database; and, how to alter or drop database objects.

Before Altering a Database

Some time after a database design has been implemented, a change to the database design may be required. You should reconsider the major design issues that you had with the previous design. You should pay particular attention to the following:

- "Changing Logical and Physical Design Characteristics"
- "Changing the License Information"
- "Changing Instances"
- "Changing Environment Variables and the Profile Registry Variables" on page 179
- "Changing the Node Configuration File" on page 179
- "Changing the Database Configuration" on page 179

Changing Logical and Physical Design Characteristics

Before you make changes affecting the entire database, you should review all the logical and physical design decisions. For example, when altering a table space, you should review your design decision regarding the use of SMS or DMS storage types. (Refer to *Administration Guide: Planning* for more information.)

Changing the License Information

As part of the management of licenses for your DB2 products, you may find that you have a need to increase the number of licenses. You can use the License Center within the Control Center to check usage of the installed products and increase the number of licenses based on that usage.

Changing Instances

Instances are designed to be as independent as possible from the effects of subsequent installation and removal of products.

In most cases, existing instances automatically inherit or lose access to the function of the product being installed or removed. However, if certain executables or components are installed or removed, existing instances do not automatically inherit the new system configuration parameters or gain access to all the additional function. The instance must be updated.

If DB2 is updated by installing a Program Temporary Fix (PTF) or a patch, all the existing DB2 instances should be updated using the **db2iupdt** command. You should also update the Administration Server (DAS) using the **dasiupdt** command.

You should ensure you understand the instances and database partition servers you have in an instance before attempting to change or delete an instance.

Listing Instances

To get a list of all the instances that are available on a system using the Control Center:

1. Expand the object tree until you see the **Databases** folder.
2. Right-click the database you want the list instances for, and select **Add** from the pop-up menu.
3. Click **Refresh**, and click the arrow at the end of the **Database name** field to see the list of instances.
4. Press **Cancel**.

To get a list of all the instances that are available on a system using the command line, enter:

```
db2ilist
```

To determine which instance applies to the current session (on OS/2 or supported Windows platforms) use:

```
set db2instance
```

Updating Instance Configuration

Running the **db2iupdt** command updates the specified instance by performing the following:

- Replaces the files in the `sqllib` subdirectory under the instance owner's home directory.
- If the node type is changed, then a new database manager configuration file is created. This is done by merging relevant values from the existing database manager configuration file with the default database manager configuration file for the new node type. If a new database manager configuration file is created, the old file is backed up to the `backup` subdirectory of the `sqllib` subdirectory under the instance owner's home directory.

The **db2iupdt** command is found in the `instance` subdirectory in the version and release subdirectory (the exact name varies by operating system).

The command is used as shown:

```
db2iupdt InstName
```

The `InstName` is the log in name of the instance owner.

There are other optional parameters associated with this command:

- –h or –?

  Displays a help menu for this command.

- –d

  Sets the debug mode for use during problem determination.

- –a AuthType

  Specifies the authentication type for the instance. Valid authentication types are SERVER, CLIENT, DCS, or DCE. If not specified, the default is SERVER, if a DB2 server is installed. Otherwise, it is set to CLIENT. The authentication type of the instance applies to all databases owned by the instance.

  On UNIX operating systems, DCE is not a valid authentication type.

- –e

  Allows you to update each instance that exists. Those that exist can be shown using **db2ilist**.

- –u FencedID

  Names the user under which the fenced user-defined functions (UDFs) and stored procedures will execute. This is not required if you install the DB2 client or the DB2 Software Developer's Kit. For other DB2 products, this is a required parameter.

  Note: FencedID may not be "root" or "bin".

- –k

  This parameter preserves the current instance type. If you do not specify this parameter, the current instance is upgraded to the highest instance type available in the following order:

  – Partitioned database server with local and remote clients (DB2 Enterprise - Extended Edition default instance type)
  – Database Server with local and remote clients (DB2 Universal Database Enterprise Edition default instance type)
  – Client (DB2 client default instance type)

Examples:

- If you installed DB2 Universal Database Workgroup Edition or DB2 Universal Database Enterprise Edition after the instance was created, enter the following command to update that instance:

```
db2iupdt -u db2fenc1 db2inst1
```

- If you installed the DB2 Connect Enterprise Edition after creating the instance, you can use the instance name as the Fenced ID also:

```
db2iupdt -u db2inst1 db2inst1
```

- To update client instances, you can use the following command:

```
db2iupdt db2inst1
```

Removing Instances

To remove an instance using the Control Center:

1. Expand the object tree until you see the instance you want to remove.
2. Right-click the instance name, and select **Remove** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To remove an instance using the command line, enter:

```
db2idrop <instance_name>
```

The preparation and details to removing an instance using the command line are:

1. Stop all applications that are currently using the instance.
2. Stop the Command Line Processor by running **db2 terminate** commands in each DB2 command window.
3. Stop the instance by running the **db2stop** command.
4. Back up the instance directory indicated by the DB2INSTPROF registry variable. On UNIX operating systems, consider backing up the files in the `INSTHOME/sqllib` directory (where `INSTHOME` is the home directory of the instance owner). For example, you might want to save the database manager configuration file, `db2systm`, the `db2nodes.cfg` file, user-defined functions (UDFs), or fenced stored procedure applications.
5. (On UNIX operating systems only) Log off as the instance owner.
6. (On UNIX operating systems only) Log in as a user with root authority.
7. Issue the **db2idrop** command:

   ```
   db2idrop InstName
   ```

   where `InstName` is the name of the instance being dropped.

   This command removes the instance entry from the list of instances and removes the instance directory.
8. (On UNIX operating systems only) Optionally, as a user with root authority, remove the instance owner's user ID and group (if used only for that instance). Do not remove these if you are planning to re-create the instance.

This step is optional since the instance owner and the instance owner group may be used for other purposes.

The **db2idrop** command removes the instance entry from the list of instances and removes the `sqllib` subdirectory under the instance owner's home directory.

Changing Environment Variables and the Profile Registry Variables

You must consider which environment variables (if any) need to be changed on your particular operating system. If any environment variables are changed and you are not on a UNIX platform, you need to restart the system for the new environment variables to take effect. Review whether you should reset the profile registry variables in the Global Profile registry before altering your database. You can then reset the profile registry variables to those that are best suited to the new database environment. If only profile registry variables have been changed, the system does not need to be restarted.

Changing the Node Configuration File

If you are planning changes to any nodegroups (both adding or deleting nodes, or moving existing nodes), you should refer to "Scaling Your Configuration Through Adding Processors" in the *Administration Guide: Performance* for details on what should be done.

Changing the Database Configuration

If you are planning changes to the database, you should review the values for the configuration parameters. Some of the values can be adjusted from time to time as part of the ongoing changes made to the database based on how it is used.

To change the database configuration, use the Performance Configuration Wizard in the Control Center. This wizard helps you tune performance and balance memory requirements for a single database per instance by suggesting which configuration parameters to modify and providing suggested values for them.

Note: If you modify any parameters, the values are not updated until:

- For database parameters, the first new connection to the database after all applications were disconnected.
- For database manager parameters, the next time you stop and start the instance.

In most cases the values recommended by the Performance Configuration Wizard will provide better performance than the default values, because they are based on information about your workload and you own particular server. However, note that the values are designed to improve the performance of,

though not necessarily optimize, your database system. Think of them as a starting point on which you can make further adjustments to obtain optimized performance.

To change the database configuration using the Control Center:

1. Expand the object tree until you see the **Databases** folder.
2. Right-click the instance or database you want to change, and select **Configure Performance Using Wizard** from the pop-up menu.
3. Click on each page and change information as required.
4. Click on the **Results** page to review you work and apply any suggested configuration parameters.
5. When you are finished applying updates, click **Finish**.

To change the database manager configuration using the command line, enter:

```
UPDATE DBM CFG FOR <database_alias>
   USING <config_keyword>=<value>
```

You can update one or more `<config_keyword>=<value>` combinations in a single command. Most changes to the database manager configuration file become effective only after they are loaded into memory. For a server configuration parameter, this occurs during the running of the START DATABASE MANAGER command. For a client configuration parameter, this occurs when the application is restarted.

To view or print the current database manager configuration parameters, use the GET DATABASE MANAGER CONFIGURATION command.

For details on how to refine your system by benchmarking, and to configure your system, refer to "Benchmark Testing" and "Configuring DB2" in the *Administration Guide: Performance*.

For multiple partitions: When you have a database that is partitioned across more than one partition, the database configuration file should be the same on all database partitions. Consistency is required since the SQL compiler compiles distributed SQL statements based on information in the node configuration file and creates an access plan to satisfy the needs of the SQL statement. Maintaining different configuration files on database partitions could lead to different access plans, depending on which database partition the statement is prepared. Use **db2_all** to maintain the configuration files across all database partitions.

Altering a Database

There are nearly as many tasks when altering databases as there are in the creation of databases. These tasks update or drop aspects of the database previously created. The tasks include:

- "Dropping a Database"
- "Altering a Nodegroup" on page 182
- "Altering a Table Space" on page 182
- "Dropping a Schema" on page 188
- "Modifying a Table in Both Structure and Content" on page 188
- "Altering a User-Defined Structured Type" on page 204
- "Deleting and Updating Rows of a Typed Table" on page 204
- "Renaming an Existing Table" on page 204
- "Dropping a Table" on page 205
- "Dropping a Trigger" on page 207
- "Dropping a User-Defined Function (UDF), Type Mapping, or Method" on page 207
- "Dropping a User-Defined Type (UDT) or Type Mapping" on page 208
- "Altering or Dropping a View" on page 209
- "Dropping a Summary Table" on page 210
- "Altering or Dropping a Server" on page 212
- "Altering or Dropping a Nickname" on page 213
- "Dropping an Index, Index Extension, or an Index Specification" on page 214
- "Statement Dependencies When Changing Objects" on page 215

Dropping a Database

Although some of the objects in a database can be altered, the database itself cannot be altered: it must be dropped and re-created. Dropping a database can have far-reaching effects, because this action deletes all its objects, containers, and associated files. The dropped database is removed (uncataloged) from the database directories.

To drop a database using the Control Center:

1. Expand the object tree until you see the **Databases** folder.
2. Right-click the database you want to drop, and select **Drop** from the pop-up menu.
3. Click on the **Confirmation** box, and click **Ok**.

To drop a database using the command line, enter:

```
DROP DATABASE <name>
```

The following command deletes the database SAMPLE:

```
DROP DATABASE SAMPLE
```

Note: If you intend to continue experimenting with the SAMPLE database, you should not drop it. If you have dropped the SAMPLE database, and find that you need it again, you can re-create it.

Altering a Nodegroup

Details on altering a nodegroup are found in the "Scaling Your Configuration Through Adding Processors" chapter in *Administration Guide: Performance*.

Once you add or drop nodes, you must redistribute the current data across the new set of nodes in the nodegroup. To do this, use the REDISTRIBUTE NODEGROUP command. For information, refer to "Redistributing Data Across Database Partitions" in the *Administration Guide: Performance* and to the *Command Reference*.

Altering a Table Space

When you create a database, you create at least three table spaces: one catalog table space (SYSCATSPACE); one user table space (with a default name of USERSPACE1); and one system temporary table space (with a default name of TEMPSPACE1). You must keep at least one of each of these table spaces. You can add additional user and temporary table spaces if you wish.

Note: You cannot drop the catalog table space SYSCATSPACE, or create another one, and there must always be at least one system temporary table space. You can create other system temporary table spaces. You also cannot change the page size or the extent size of a table space after it has been created.

This section discusses how to change table spaces as follows:

- "Adding a Container to a DMS Table Space"
- "Modifying Containers in a DMS Table Space" on page 183
- "Adding a Container to an SMS Table Space on a Partition" on page 185
- "Renaming a Table Space" on page 185
- "Switching the State of a Table Space" on page 185
- "Dropping a User Table Space" on page 186
- "Dropping a System Temporary Table Space" on page 186.

Refer to *Administration Guide: Planning* for design information on table spaces.

Adding a Container to a DMS Table Space

You can increase the size of a DMS table space (that is, one created with the MANAGED BY DATABASE clause) by adding one or more containers to the table space.

The contents of the table space are re-balanced across all containers. Access to the table space is not restricted during the re-balancing. If you need to add more than one container, you should add them at the same time.

To add a container to a DMS table space using the Control Center:

1. Expand the object tree until you see the **Table Spaces** folder.
2. Right-click the table space where you want to add the container, and select **Alter** from the pop-up menu.
3. Click **Add**, complete the information, and click **Ok**.
4. If the table space is in a partitioned database environment, click **Advanced** if you need to change performance parameters for the table space.
5. Click **Ok**.

To add a container to a DMS table space using the command line, enter:

```
ALTER TABLESPACE <name>
   ADD (DEVICE '<path>' <size>)
```

The following example illustrates how to add two new device containers (each with 10 000 pages) to a table space on a UNIX-based system:

```
ALTER TABLESPACE RESOURCE
  ADD (DEVICE '/dev/rhd9'  10000,
       DEVICE  '/dev/rhd10' 10000)
```

Note that the ALTER TABLESPACE statement allows you to change other properties of the table space that can affect performance. Refer to "Table Space Impact on Query Optimization" in the *Administration Guide: Performance* for more information.

Modifying Containers in a DMS Table Space

You can increase the size of the containers in a DMS table space (that is, one created with the MANAGED BY DATABASE clause) by resizing one or more containers or by extending one or more containers associated with the table space. You should consider the resizing method if you know the new upper limit for the size of the container. You should consider the extend method if you do not know (nor care about) the current size of the container.

To resize one or more containers in a DMS table space using the command line, enter:

```
ALTER TABLESPACE <name>
   RESIZE (DEVICE '<path>' <size>)
```

The following example illustrates how to increase two device containers (each already existing with 1 000 pages) in a table space on a UNIX-based system:

```
ALTER TABLESPACE HISTORY
  RESIZE (DEVICE '/dev/rhd7'  2000,
          DEVICE  '/dev/rhd8' 2000)
```

Following this action, the two devices have increased from 1 000 pages in size to 2 000 pages. Similar to adding new containers, the contents of the table space are re-balanced across all containers. Access to the table space is not restricted during the re-balancing.

To extend one or more containers in a DMS table space using the command line, enter:

```
ALTER TABLESPACE <name>
  EXTEND (DEVICE '<path>' <size>)
```

The following example illustrates how to increase two device containers (each already existing with 1 000 pages) in a table space on a UNIX-based system:

```
ALTER TABLESPACE HISTORY
  EXTEND (DEVICE '/dev/rhd11'  1000,
          DEVICE  '/dev/rhd12' 1000)
```

Following this action, the two devices have increased from 1 000 pages in size to 2 000 pages. Similar to adding new containers, the contents of the table space are re-balanced across all containers. Access to the table space is not restricted during the re-balancing.

DMS containers (both file and raw device containers) which are added during or after table space creation, or are extended after table space creation, are performed in parallel through prefetchers. To achieve an increase in parallelism of these create or resize container operations, you can increase the number of prefetchers running in the system. The only process which is not done in parallel is the logging of these actions and, in the case of creating containers, the tagging of the containers.

Note: To maximize the parallelism of the CREATE TABLESPACE or ALTER TABLESPACE statements (with respect to adding new containers to an existing table space) ensure the number of prefetchers is greater than or equal to the number of containers being added.

Note: You cannot reduce the size of the containers.

Note that the ALTER TABLESPACE statement allows you to change other properties of the table space that can affect performance. Refer to "Table Space Impact on Query Optimization" in the *Administration Guide: Performance* for more information.

Adding a Container to an SMS Table Space on a Partition

To add a container to an SMS table space using the command line, enter the following:

```
ALTER TABLESPACE <name>
   ADD ('<path>')
   ON NODE (<partition_number>)
```

The partition specified by number, and every partition (or node) in the range of partitions, must exist in the nodegroup on which the table space is defined. A partition_number may only appear explicitly or within a range in exactly one *on-nodes-clause* for the statement.

The following example shows how to add a new container to partition number 3 of the nodegroup used by table space "plans" on a UNIX based operating system:

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON NODE (3)
```

Renaming a Table Space

You can give an existing table space a new name without being concerned with the individual objects within the table space. When renaming a table space, all the catalog records referencing that table space are changed.

You cannot rename the SYSCATSPACE table space.

You cannot rename a table space that is in a "roll-forward pending" or "roll-forward in progress" state.

When restoring a table space that has been renamed since it was backed up, you must use the new table space name in the RESTORE DATABASE command. If you use the previous table space name, it will not be found. Similarly, if you are rolling forward the table space with the ROLLFORWARD DATABASE command, ensure that you use the new name. If the previous table space name is used, it will not be found.

Switching the State of a Table Space

The SWITCH ONLINE clause of the ALTER TABLESPACE statement can be used to remove the OFFLINE state from a table space if the containers associated with that table space have become accessible. The table space has the OFFLINE state removed while the rest of the database is still up and being used.

An alternative to the use of this clause is to disconnect all applications from the database and then to have the applications connect to the database again. This removes the OFFLINE state from the table space.

To remove the OFFLINE state from a table space using the command line, enter:

```
ALTER TABLESPACE <name>
   SWITCH ONLINE
```

Dropping a User Table Space

When you drop a user table space, you delete all the data in that table space, free the containers, remove the catalog entries, and cause all objects defined in the table space to be either dropped or marked as invalid.

You can reuse the containers in an empty table space by dropping the table space, but you must COMMIT the DROP TABLESPACE command before attempting to reuse the containers.

You can drop a user table space that contains all of the table data including index and LOB data within that single user table space. You can also drop a user table space that may have tables spanned across several table spaces. That is, you may have table data in one table space, indexes in another, and any LOBs in a third table space. You must drop all three table spaces at the same time in a single statement. All of the table spaces that contain tables that are spanned must be part of this single statement or the drop request will fail. Refer to the *SQL Reference* for details on how to drop table spaces containing spanned table data.

To drop a user table space using the Control Center:

1. Expand the object tree until you see the **Table Spaces** folder.
2. Right-click on the table space you want to drop, and select **Drop** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To drop a user table space using the command line, enter:

```
DROP TABLESPACE <name>
```

The following SQL statement drops the table space ACCOUNTING:

```
DROP TABLESPACE ACCOUNTING
```

Dropping a System Temporary Table Space

You cannot drop a system temporary table space without first creating another system temporary table space because the database must always have at least one system temporary table space. For example, if you wish to add a container to an SMS temporary table space, you must add a new system temporary table space first and then drop the old system temporary table space.

To drop a system table space using the Control Center:

1. Expand the object tree until you see the **Table Spaces** folder.
2. If there is only one other system temporary table space, right-click the **Table Spaces** folder, and select **Create —> Table Space Using Wizard** from the pop-up menu. Otherwise, skip to step four.
3. Follow the steps in the wizard to create the new system temporary table space if needed.
4. Click again on the **Table Spaces** folder to display a list of table spaces in the right side of the window (the Contents pane).
5. Right-click on the system temporary table space you want to drop, and click **Drop** from the pop-up menu.
6. Check the **Confirmation** box, and click **Ok**.

If you only have one system temporary table space, before deleting it, you must create another. This can be done using the command line by entering:

```
CREATE SYSTEM TEMPORARY TABLESPACE <name>
   MANAGED BY SYSTEM USING ('<device>')
```

Then, to drop a system table space using the command line, enter:

```
DROP TABLESPACE <name>
```

The following SQL statement creates a new system temporary table space called TEMPSPACE2:

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
   MANAGED BY SYSTEM USING ('d')
```

Once TEMPSPACE2 is created, you can then drop the original system temporary table space TEMPSPACE1 with the command:

```
DROP TABLESPACE TEMPSPACE1
```

You can reuse the containers in an empty table space by dropping the table space, but you must COMMIT the DROP TABLESPACE command before attempting to reuse the containers.

Dropping a User Temporary Table Space

You can only drop a user temporary table space if there are no current declared temporary tables defined in that table space. When you drop the table space, no attempt is made to drop all of the declared temporary tables in the table space.

Note: A declared temporary table is implicitly dropped when the application that declared it disconnects from the database.

Dropping a Schema

Before dropping a schema, all objects that were in that schema must be dropped themselves or moved to another schema. The schema name must be in the catalog when attempting the DROP statement; otherwise an error is returned.

To drop a schema using the Control Center:

1. Expand the object tree until you see the **Schemas** folder.
2. Right-click on the schema you want to drop, and select **Drop** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To drop a schema using the command line, enter:

```
DROP SCHEMA <name>
```

In the following example, the schema ″joeschma″ is dropped:

```
DROP SCHEMA joeschma RESTRICT
```

The `RESTRICT` keyword enforces the rule that no objects can be defined in the specified schema for the schema to be deleted from the database.

Modifying a Table in Both Structure and Content

Tasks that are required for modifying the structure and content of the table include the following:

- "Adding Columns to an Existing Table"
- "Modifying a Column Definition" on page 189
- "Removing Rows From a Table or View" on page 190
- "Altering a Constraint" on page 191
- "Defining a Generated Column on an Existing Table" on page 196
- "Declaring a Table Volatile" on page 199
- "Changing Partitioning Keys" on page 200
- "Changing Table Attributes" on page 201
- "Refreshing the Data in a Summary Table" on page 204

Note that you cannot alter triggers for tables; you must drop any trigger that is no longer appropriate (see "Dropping a Trigger" on page 207), and add its replacement (see "Creating a Trigger" on page 136).

Adding Columns to an Existing Table

A column definition includes a column name, data type, and any necessary constraints.

When columns are added to a table, the columns are logically placed to the right of the right-most existing column definition. When a new column is added to an existing table, only the table description in the system catalog is modified, so access time to the table is not affected immediately. Existing records are not physically altered until they are modified using an UPDATE statement. When retrieving an existing row from the table, a null or default value is provided for the new column, depending on how the new column was defined. Columns that are added after a table is created cannot be defined as NOT NULL: they must be defined as either NOT NULL WITH DEFAULT or as nullable.

To add columns to an existing table using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to add columns to, and select **Alter** from the pop-up menu.
3. Check the **Columns** page, complete the information for the column, and click **Ok**.

To add columns to an existing table using the command line, enter:

```
ALTER TABLE <table_name>
   ADD <column_name> <data_type> <null_attribute>
```

Columns can be added with an SQL statement. The following statement uses the ALTER TABLE statement to add three columns to the EMPLOYEE table:

```
ALTER TABLE EMPLOYEE
   ADD MIDINIT CHAR(1)   NOT NULL WITH DEFAULT
   ADD HIREDATE DATE
   ADD WORKDEPT CHAR(3)
```

Modifying a Column Definition

You can modify the characteristics of a column by increasing the length of an existing VARCHAR column. The number of characters may increase up to a value dependent on the page size used.

To modify columns of an existing table using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. In the list of tables in the right pane, right-click on the table for which you want to modify a column, and select **Alter** from the pop-up menu.
3. Check the **Columns** page, select the column, and click **Change**.
4. Type the new byte count for the column in **Length**, and click **Ok**.

To modify columns of an existing table using the command line, enter:

```
ALTER TABLE ALTER COLUMN
   <column_name> <modification_type>
```

For example, to increase a column up to 4000 characters, use something similar to the following:

```
ALTER TABLE ALTER COLUMN
   COLNAM1 SET DATA TYPE VARCHAR(4000)
```

You cannot alter the column of a typed table. However, you can add a scope to an existing reference type column that does not already have a scope defined. For example:

```
ALTER TABLE ALTER COLUMN
   COLNAMT1 ADD SCOPE TYPTAB1
```

For more information about the ALTER TABLE statement, refer to the *SQL Reference* manual.

Removing Rows From a Table or View

You can change the contents of a table or view by deleting rows. Deleting a row from a view deletes the rows from the table on which the view is based. The DELETE statement is used to:

- Delete one or more rows that have been optionally determined by a search condition. This is known as a *searched DELETE*.
- Delete exactly one row that has been determined by the current position of a cursor. This is known as a *positioned DELETE*.

The DELETE statement can be embedded in an application program or issued as a dynamic SQL statement.

If the table being modified is involved with other tables through referential constraints then there are considerations with carrying out the deletion of rows. If the identified table or the base table of the identified view is a parent, the rows selected for delete must not have any dependents in a relationship with a delete rule of RESTRICT. Further, the DELETE must not cascade to descendent rows that have dependents in a relationship with a delete rule of RESTRICT.

If the delete operation is not prevented by a RESTRICT delete rule, the selected rows are deleted. For additional information concerning what happens to the rows that are dependents of the selected rows, you should refer to the *SQL Reference*.

For example, to delete the department (DEPTNO) "D11" from the table (DEPARTMENT), use:

```
DELETE FROM department WHERE deptno='D11'
```

If an error occurs during the running of a multiple row DELETE, no changes are made to the table. If an error occurs that prevents deleting all rows matching the search condition and all operations required by existing referential constraints, no changes are made to the tables.

Unless appropriate locks already exist, one or more exclusive locks are acquired during the running of a successful DELETE statement. Locks are released following a COMMIT or ROLLBACK statement. Locks can prevent other applications from performing operations on the table.

Modifying an Identity Column Definition

If you are recreating a table followed by an import or load operation, and if you have an IDENTITY column in the table then it will be reset to start generating the IDENTITY value from 1 following the recreation of the contents of the table. When inserting new rows into this recreated table, you do not want the IDENTITY column to begin from 1 again. You do not want duplicate values in the IDENTITY column. To prevent this from occuring, you should:

1. Recreate the table.
2. Load data into the table using the MODIFIED BY IDENTITYOVERRIDE clause. The data is loaded into the table but no identity values are generated for the rows.
3. Run a query to get the last counter value for the IDENTITY column:

   ```
   SELECT MAX(<IDENTITY column>)
   ```

   This will return with the equivalent value of what would have been the IDENTITY column value of the table.
4. Use the RESTART clause of the ALTER TABLE statement:

   ```
   ALTER TABLE <table name> ALTER COLUMN <IDENTITY column>
      RESTART WITH <last counter value>
   ```
5. Insert a new row into the table. The IDENTITY column value will be generated based on the value specified in the RESTART WITH clause.

Altering a Constraint

You can only alter constraints by dropping them and then adding new ones to take their place. For more information, see:

- "Adding a Constraint"
- "Dropping a Constraint" on page 194

For more information on constraints, see "Defining Constraints" on page 123.

Adding a Constraint

You add constraints with the ALTER TABLE statement. For more information on this statement, including its syntax, refer to the *SQL Reference* manual.

For more information on constraints, see "Defining Constraints" on page 123.

Adding a Unique Constraint: Unique constraints can be added to an existing table. The constraint name cannot be the same as any other constraint specified within the ALTER TABLE statement, and must be unique within the table (this includes the names of any referential integrity constraints that are defined). Existing data is checked against the new condition before the statement succeeds.

The following SQL statement adds a unique constraint to the EMPLOYEE table that represents a new way to uniquely identify employees in the table:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

Adding Primary and Foreign Keys: 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 violating 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.

To add primary keys using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Primary Key** page, select one or more columns as primary keys, and click the arrow to move them.
4. Optional: Enter the constraint name of the primary key.
5. Click **Ok**.

To add primary keys using the command line, enter:

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
   PRIMARY KEY <column_name>
```

When a foreign key is added to a table, packages and cached dynamic SQL containing the following statements may be marked as invalid:

- Statements that insert or update the table containing the foreign key
- Statements that update or delete the parent table.

See "Statement Dependencies When Changing Objects" on page 215 for information.

To add foreign keys using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Foreign Keys** page, click **Add**.
4. On the **Add Foreign Keys** window, specify the parent table information.
5. Select one or more columns to be foreign keys, and click the arrow to move them.
6. Specify what action is to take place on the dependent table when a row of the parent table is deleted or updated. You can also add a constraint name for he foreign key.
7. Click **Ok**.

To add foreign keys using the command line, enter:

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
    FOREIGN KEY <column_name>
    ON DELETE <action_type>
    ON UPDATE <action_type>
```

The following examples show the ALTER TABLE statement to add primary keys and foreign keys to a table:

```
ALTER TABLE PROJECT
  ADD CONSTRAINT PROJECT_KEY
      PRIMARY KEY (PROJNO)
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
      PRIMARY KEY (EMPNO, PROJNO, ACTNO)
  ADD CONSTRAINT ACT_EMP_REF
      FOREIGN KEY (EMPNO)
      REFERENCES EMPLOYEE
      ON DELETE  RESTRICT
  ADD CONSTRAINT ACT_PROJ_REF
      FOREIGN KEY (PROJNO)
      REFERENCES PROJECT
      ON DELETE CASCADE
```

Adding a Table Check Constraint: Check constraints can be added to an existing table with the ALTER TABLE statement. The constraint name cannot be the same as any other constraint specified within an ALTER TABLE statement, and must be unique within the table (this includes the names of any referential integrity constraints that are defined). Existing data is checked against the new condition before the statement succeeds.

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 violating 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 and cached dynamic SQL that insert or update the table may be marked as invalid. See "Statement Dependencies When Changing Objects" on page 215 for more information.

To add a table check constraint using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Check Constraints** page, click **Add**.
4. On the **Add Check Constraint** window, complete the information, and click **Ok**.
5. On the **Check Constraints** page, click **Ok**.

To add a table check constraint using the command line, enter:

```
ALTER TABLE <name>
   ADD CONSTRAINT <name> (<constraint>)
```

The following SQL statement adds a constraint to the EMPLOYEE table that the salary plus commission of each employee must be more than $25,000:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

Dropping a Constraint

You drop constraints with the ALTER TABLE statement. For more information on this statement, including its syntax, refer to the *SQL Reference* manual.

For more information on constraints, see "Defining Constraints" on page 123.

Dropping a Unique Constraint: You can explicitly drop a unique constraint using the ALTER TABLE statement. The name of all unique constraints on a table can be found in the SYSCAT.INDEXES system catalog view.

The following SQL statement drops the unique constraint NEWID from the EMPLOYEE table:

```
ALTER TABLE EMPLOYEE
   DROP UNIQUE NEWID
```

Dropping this unique constraint invalidates any packages or cached dynamic SQL that used the constraint.

Dropping Primary and Foreign Keys: To drop a primary key using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Primary Key** page, select the primary key at right to drop, and click the arrow to move it to the **Available columns** box on the left.
4. Click **Ok**.

To drop a primary key using the command line, enter:

```
ALTER TABLE <name>
   DROP PRIMARY KEY
```

When a foreign key constraint is dropped, packages or cached dynamic SQL statements containing the following may be marked as invalid:

- Statements that insert or update the table containing the foreign key
- Statements that update or delete the parent table.

See "Statement Dependencies When Changing Objects" on page 215 for more information.

To drop a foreign key using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Foreign Keys** page, click **Add**.
4. Select the foreign key at right to drop, and click on the arrow to move it to the **Available columns** box on the left.
5. On the **Foreign Keys** page, click **Ok**.

To drop a foreign key using the command line, enter:

```
ALTER TABLE <name>
   DROP FOREIGN KEY <foreign_key_name>
```

The following examples use the DROP PRIMARY KEY and DROP FOREIGN KEY clauses in the ALTER TABLE statement to drop primary keys and foreign keys on a table:

```
ALTER TABLE EMP_ACT
  DROP PRIMARY KEY
  DROP FOREIGN KEY ACT_EMP_REF
  DROP FOREIGN KEY ACT_PROJ_REF
ALTER TABLE PROJECT
  DROP PRIMARY KEY
```

For information about the ALTER TABLE statement, refer to the *SQL Reference* manual.

Dropping a Table Check Constraint: You can explicitly drop or change a table check constraint using the ALTER TABLE statement, or implicitly drop it as the result of a DROP TABLE statement.

When you drop a table check constraint, all packages and cached dynamic SQL statements with INSERT or UPDATE dependencies on the table are invalidated. (See "Statement Dependencies When Changing Objects" on page 215 for more information.) The name of all check constraints on a table can be found in the SYSCAT.CHECKS catalog view. Before attempting to drop a table check constraint having a system-generated name, look for the name in the SYSCAT.CHECKS catalog view.

To drop a table check constraint using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Check Constraints** page, select the check constraint to drop, click **Remove**, and click **Ok**.

To drop a table check constraint using the command line:

```
ALTER TABLE <table_name>
   DROP CHECK <check_constraint_name>
```

The following SQL statement drops the table check constraint REVENUE from the EMPLOYEE table:

```
ALTER TABLE EMPLOYEE
   DROP CHECK REVENUE
```

Defining a Generated Column on an Existing Table

A generated column is defined on a base table where the stored value is computed using an expression, rather than being specified through an insert or update operation. A generated column can be created when a table is created or as a modification to an existing table.

Perform the following steps to define a generated column:

1. Place the table in a check pending state.

   ```
   SET INTEGRITY FOR t1 OFF
   ```

2. Alter the table to add one or more generated columns.

   ```
   ALTER TABLE t1 ADD COLUMN c3 DOUBLE GENERATED ALWAYS AS (c1 + c2),
                  ADD COLUMN c4 GENERATED ALWAYS AS
                     (CASE WHEN c1 > c3 THEN 1 ELSE NULL END))
   ```

3. At this point there are several ways to complete this task based on the work to be done against the table:
   - The table is very large and you are not confident that you have sufficient log space to complete the task. After loading the data but before you turn the integrity checking back on, you need to:

     ```
     COMMIT
     ```

     Then you need to use the db2gncol utility to establish the values for the generated columns. This utility is located under the `sqllib` directory in the `bin` subdirectory. You use the utility as follows:

     ```
     db2gncol -d <dbname> -s <schema> -t <table_name>
              -c <commitcount>
     ```

     The `dbname` specifies an alias name for the database in which the table is located. The `schema` specifies the schema name of the table and is case sensitive. The `table_name` specifies the table for which new values for its columns generated by expressions are to be computed. Both `schema` and `table_name` are case-sensitive. The `commitcount` is the number of rows to process between each internal commit to clean up the logs. This parameter influences the size of the log space required to perform the generation of the column values.

     There are also two optional parameters that are not shown in the example above. They are `-u <username>` and `-p <password>` which identify a user and password. The user must have SYSADM or DBADM authority. If there is no user and password identified, the current user ID will be used.

     If you wish help information on this utility, enter:

     ```
     db2gncol -h
     ```

     When the help parameter is used, all other parameters are ignored.

     The table is locked for the entire process even though it is in a check pending state. The reason for the lock is that there are other utilities that can access tables that are in a check pending state. The lock prevents conflicts with these other utilities.
   - You anticipate that the log space for updating the generated columns is expected to be sufficient for SET INTEGRITY. This will normally be the case. After loading the data you recompute and reassign the values for the generated columns using:

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED
        FORCE GENERATED
     ```

     Note: Exception tables can be used at this point.

- The table is very large, you are not confident you will have sufficient log space to complete the task, and you do not choose the first method presented above. After loading the data but before you turn the integrity checking back on, you need to:
  a. Get an exclusive lock on the table. This prevents all but uncommitted read transactions from accessing the table.
     ```
     LOCK TABLE t1
     ```
  b. Move the table to the online state with the data unchecked.
     ```
     SET INTEGRITY FOR t1 ALL IMMEDIATE UNCHECKED
     ```
  c. Update the generated columns using intermittent commits and predicates to avoid the logs filling up.
     ```
     UPDATE t1 SET (c3, c4) = (DEFAULT, DEFAULT) WHERE <predicate>
     ```
  d. Bring up the table online and check its integrity.
     ```
     SET INTEGRITY FOR t1 OFF
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED
     ```
  e. Unlock the table by completing the transaction using a commit statement.
     ```
     COMMIT
     ```
- You know that the table was created with the not logged initially option. In this way, logging for the table is turned off with the usual implications and risks while working with the generated column values.
  a. Activate the not logged initially option.
     ```
     ALTER TABLE t1 ACTIVATE NOT LOGGED INITIALLY
     ```
  b. Generate the values.
     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATED
     ```
  c. Turn the not logged initially off again by committing the transaction.
     ```
     COMMIT
     ```

The values for generated columns can also simply be checked by applying the expression as if it is an equality check constraint:

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

If values have been placed in a generated column using LOAD for example, and you know that the values match the generated expression, then the table can be taken out of the check pending state without checking or assigning the values:

```
SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
```

Generated columns may only be defined on data types for which an equal comparison is defined. The excluded data types for the generated columns

include: Structured types, LOBs, CLOBs, DBCLOBs, LONG VARCHAR, LONG VARGRAPHIC, and user-defined types defined using the same excluded data types.

Generated columns cannot be used in constraints, unique indexes, referential constraints, primary keys, and global temporary tables. A table created with LIKE and materialized views does not inherit generated column properties.

Generated columns cannot be inserted or updated without the keyword DEFAULT. When inserting, the use of DEFAULT avoids the need to enumerate the columns in the column list. Instead, generated columns can be set to DEFAULT in the values list. When updating, DEFAULT enables the recomputation of generated columns that have been placed online by SET INTEGRITY without being checked.

The order of processing of triggers requires that BEFORE-triggers may not reference generated columns in their header (before update) or in their bodies. In the order of processing, generated columns are processed after BEFORE-triggers.

The db2look utility will not see the check constraints generated by a generated column.

When using replication, the target table must not use generated columns in its mapping. There are two choices when replicating:

- The target table must define the generated column as a normal column; that is, not a generated column
- The target table must omit the generated column in the mapping

There are several restrictions when working with generated columns:

- Generated columns must not have dependencies on each other.
- The expressions used to create the generated columns must not contain subqueries. This includes expressions with functions that READS SQL DATA.
- No check constraints are allowed on generated columns.
- No unique indexes are allowed on generated columns. This includes unique constraints and primary keys.

Declaring a Table Volatile

A *volatile* table is defined as a table whose contents can vary from empty to very large at run time. The volatility or extreme changeability of this type of table makes reliance on the statistics collected by RUNSTATS inaccurate. Statistics are gathered at, and only reflect, a point in time. To generate an access plan that uses a volatile table can result in an incorrect or poorly

performing plan. For example, if statistics are gathered when the volatile table is empty, the optimizer tends to favor accessing the volatile table using a table scan rather than an index scan.

To prevent this, you should consider declaring the table as volatile using the ALTER TABLE statement. By declaring the table volatile, the optimizer will consider using index scan rather than table scan. The access plans that use declared volatile tables will not depend on the existing statistics for that table.

To declare a table volatile using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Table** page, select the **Cardinality varies significantly at run time** check box, and click **Ok**.

To declare a table as "volatile" using the command line, enter:

```
ALTER TABLE <table_name>
   VOLATILE CARDINALITY
```

Changing Partitioning Keys

You can only change a partitioning key on tables in single-partition nodegroups. First drop the existing partitioning key, and then create another.

The following SQL statement drops the partitioning key MIX_INT from the MIXREC table:

```
ALTER TABLE MIXREC
   DROP PARTITIONING KEY
```

For more information, see the ALTER TABLE statement in the *SQL Reference* manual.

You cannot change the partitioning key of a table in a multiple partition nodegroup. If you try to drop it, an error is returned.

To change the partitioning key of multiple partition nodegroups, either:

- Export all of the data to a single-partition nodegroup and then follow the above instructions.
- Export all of the data, drop the table, recreate the table redefining the partitioning key, and then import all of the data.

Neither of these methods are practical for large databases; it is therefore essential that you define the appropriate partitioning key before implementing the design of large databases.

Changing Table Attributes

You may have reason to change table attributes such as the data capture option, the percentage of free space on each page (PCTFREE), the lock size, or the append mode.

The amount of free space to be left on each page of a table is specified through PCTFREE, and is an important consideration for the effective use of clustering indexes. The amount to specify depends on the nature of the existing data and expected future data. PCTFREE is respected by LOAD and REORG but is ignored by insert, update and import activities.

Setting PCTFREE to a larger value will maintain clustering for a longer period, but will also require more disk space.

You can specify the size (granularity) of locks used when the table is accessed by using the LOCKSIZE parameter. By default, when the table is created, row level locks are defined. Use of table level locks may improve the performance of queries by limiting the number of locks that need to be acquired and released.

By specifying APPEND ON, you can improve the overall performance of the table. It allows for faster insertions, while eliminating the maintenance of information about the free space.

A table with a clustering index cannot be altered to have append mode turned on. Similarly, a clustering index cannot be created on a table with append mode.

Altering an Identity Column

Modify the attributes of an existing identity column with the ALTER TABLE statement. For more information on this statement, including its syntax, refer to the *SQL Reference*.

There are several ways to modify an identity column so that it has some of the characteristics of sequences.

There are some tasks that are unique to the ALTER TABLE statement and the identity column:

- RESTART resets the sequence associated with the identity column to the value specified implicitly or explicitly as the starting value when the identity column was originally created.
- RESTART WITH <numeric-constant> resets the sequence associated with the identity column to the exact numeric constant value. The numeric constant could be any positive or negative value with no non-zero digits to the right of any decimal point that could be assigned to the identity column.

Altering a Sequence

Modify the attributes of an existing sequence with the ALTER SEQUENCE statement. For more information on this statement, including its syntax, refer to the *SQL Reference*.

The attributes of the sequence that can be modified include:

- Changing the increment between future values
- Establishing new minimum or maximum values
- Changing the number of cached sequence numbers
- Changing whether the sequence will cycle or not
- Changing whether sequence numbers must be generated in order of request
- Restarting the sequence

There are two tasks that are not found as part of the creation of the sequence. They are:

- RESTART. Resets the sequence to the value specified implicitly or explicitly as the starting value when the sequence was created.
- RESTART WITH numeric-constant. Resets the sequence to the exact numeric constant value. The numeric constant can be any positive or negative value with no non-zero digits to the right of any decimal point.

After restarting a sequence or changing to CYCLE, it is possible to generate duplicate sequence numbers. Only future sequence numbers are affected by the ALTER SEQUENCE statement.

The data type of a sequence cannot be changed. Instead, you must drop the current sequence and then create a new sequence specifying the new data type.

All cached sequence values not used by DB2 are lost when a sequence is altered.

Dropping a Sequence

To delete a sequence, use the DROP statement. For more information on this statement, including its syntax, refer to the *SQL Reference*.

A specific sequence can be dropped by using:

```
DROP SEQUENCE sequence_name
```

where the `sequence_name` is the name of the sequence to be dropped and includes the implicit or explicit schema name to exactly identify an existing sequence.

Sequences that are system-created for IDENTITY columns cannot be dropped using the DROP SEQUENCE statement.

Once a sequence is dropped, all privileges on the sequence are also dropped.

Altering Summary Table Properties

With some restrictions, you can change a summary table to a regular table or a regular table to a summary table. You cannot change other table types; only regular and summary tables can be changed. For example, you cannot change a replicated summary table to a regular table, nor the reverse.

Once a regular table has been altered to a summary table, the table is placed in a check pending state. When altering in this way, the fullselect in the summary table definition must match the original table definition, that is:

- The number of columns must be the same.
- The column names and positions must match.
- The data types must be identical.

If the summary table is defined on an original table, then the original table cannot itself be altered into a summary table. If the original table has triggers, check constraints, referential constraints, or a defined unique index, then it cannot be altered into a summary table. If altering the table properties to define a summary table, you are not allowed to alter the table in any other way in the same ALTER TABLE statement.

When altering a regular table into a summary table, the fullselect of the summary table definition cannot reference the original table directly or indirectly through views, aliases, or summary tables.

To change a summary table to a regular table, use the following:

```
ALTER TABLE sumtable
   SET SUMMARY AS DEFINITION ONLY
```

To change a regular table to a summary table, use the following:

```
ALTER TABLE regtable
   SET SUMMARY AS <fullselect>
```

The restrictions on the fullselect when altering the regular table to a summary table are very much like the restrictions when creating a summary table using the CREATE SUMMARY TABLE statement.

Refer to the *SQL Reference* for additional information on the CREATE SUMMARY TABLE statement.

Refreshing the Data in a Summary Table

You can refresh the data in one or more summary tables by using the REFRESH TABLE statement. The statement can be embedded in an application program, or issued dynamically. To use this statement, you must have either SYSADM or DBADM authority, or CONTROL privilege on the table to be refreshed.

The following example shows how to refresh the data in a summary table:

```
REFRESH TABLE SUMTAB1
```

For more information about the REFRESH TABLE statement, refer to the *SQL Reference*.

Altering a User-Defined Structured Type

After creating a structured type, you may find that you need to add or drop attributes associated with that structured type. This is done using the ALTER TYPE (Structured) statement. Refer to the *Application Development Guide* for all the information you need on structured types.

Deleting and Updating Rows of a Typed Table

Rows can be deleted from typed tables using either searched or positioned DELETE statements. Rows can be updated in typed tables using either searched or positioned UPDATE statements. Refer to the *Application Development Guide* for all the information you need on typed tables.

Renaming an Existing Table

You can give an existing table a new name within a schema and maintain the authorizations and indexes that were created on the original table.

The existing table to be renamed can be an alias identifying a table. The existing table to be renamed must not be the name of a catalog table, a summary table, a typed table, or an object other than a table or an alias.

The existing table cannot be referenced in any of the following:

- Views
- Triggers
- Referential constraints
- Summary table
- The scope of an existing reference column

Also, there must be no check constraints within the table nor any generated columns other than the identity column. Any packages or cached dynamic SQL statements dependent on the original table are invalidated. Finally, any aliases referring to the original table are not modified.

You should consider checking the appropriate system catalog tables to ensure that the table being renamed is not affected by any of these restrictions.

Packages must be re-bound if they refer to a table that has just been renamed. The packages can be implicitly re-bound if:

- Another table is renamed using the original name of the table, or
- An alias or view is created using the original name of the table.

One of these two choices must be completed before any implicit or explicit re-binding is attempted. If neither choice is made, any re-bind will fail.

To rename an existing table using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to rename, and select **Rename** from the pop-up menu.
3. Type the new table name, and click **Ok**.

To rename an existing table using the command line, enter:

```
RENAME TABLE <schema_name>.<table_name> TO <new_name>
```

The SQL statement below renames the EMPLOYEE table within the COMPANY schema to EMPL:

```
RENAME TABLE COMPANY.EMPLOYEE TO EMPL
```

For more information about the RENAME TABLE statement, refer to the *SQL Reference* manual.

Dropping a Table

A table can be dropped with a DROP TABLE SQL statement.

When a table is dropped, the row in the SYSCAT.TABLES catalog that contains information about that table is dropped, and any other objects that depend on the table are affected. For example:

- All column names are dropped.
- Indexes created on any columns of the table are dropped.
- All views based on the table are marked inoperative. (See "Recovering Inoperative Views" on page 210 for more information.)
- All privileges on the dropped table and dependent views are implicitly revoked.
- All referential constraints in which the table is a parent or dependent are dropped.

- All packages and cached dynamic SQL statements dependent on the dropped table are marked invalid, and remain so until the dependent objects are re-created. This includes packages dependent on any supertable above the subtable in the hierarchy that is being dropped. (See "Statement Dependencies When Changing Objects" on page 215 for more information.)
- Any reference columns for which the dropped table is defined as the scope of the reference become "unscoped".
- An alias definition on the table is not affected, because an alias can be undefined
- All triggers dependent on the dropped table are marked inoperative.
- All files that are linked through any DATALINK columns are unlinked. The unlink operation is performed asynchronously which means the files may not be immediately available for other operations.

To drop a table using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the table you want to drop, and select **Drop** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To drop a table using the command line, enter:

```
DROP TABLE <table_name>
```

The following statement drops the table called DEPARTMENT:

```
DROP TABLE DEPARTMENT
```

An individual table cannot be dropped if it has a subtable. However, all the tables in a table hierarchy can be dropped by a single DROP TABLE HIERARCHY statement, as in the following example:

```
DROP TABLE HIERARCHY person
```

The DROP TABLE HIERARCHY statement must name the root table of the hierarchy to be dropped.

There are differences when dropping a table hierarchy compared to dropping a specific table:

- DROP TABLE HIERARCHY does not activate deletion-triggers that would be activated by individual DROP table statements. For example, dropping an individual subtable would activate deletion-triggers on its supertables.
- DROP TABLE HIERARCHY does not make log entries for the individual rows of the dropped tables. Instead, the dropping of the hierarchy is logged as a single event.

Refer to the *SQL Reference* for more information on the DROP statement.

Dropping a User-Defined Temporary Table

There are a few considerations to be noted when dropping a user-defined temporary table; that is, one created using the DECLARE GLOBAL TEMPORARY TABLE statement.

When dropping such a table, the table name must be qualified by the schema name SESSION and must exist in the application that created the table.

Packages cannot be dependent on this type of table and therefore they are not invalidated when such a table is dropped.

When a user-defined temporary table is dropped, and its creation preceded the active unit of work or savepoint, then the table is functionally dropped and the application is not able to access the table. However, the table still has some space reserved in its table space and this prevents the user temporary table space from being dropped until the unit of work is committed or the savepoint is ended.

Refer to the *SQL Reference* for more information on the DROP statement.

Dropping a Trigger

A trigger object can be dropped using the DROP statement, but this procedure will cause dependent packages to be marked invalid, as follows:

- If an update trigger without an explicit column list is dropped, then packages with an update usage on the target table are invalidated.
- If an update trigger with a column list is dropped, then packages with update usage on the target table are only invalidated if the package also had an update usage on at least one column in the `column-name` list of the CREATE TRIGGER statement.
- If an insert trigger is dropped, packages that have an insert usage on the target table are invalidated.
- If a delete trigger is dropped, packages that have a delete usage on the target table are invalidated.

A package remains invalid until the application program is explicitly bound or rebound, or it is run and the database manager automatically rebinds it.

Dropping a User-Defined Function (UDF), Type Mapping, or Method

A user-defined function (UDF), function template, or function mapping can be dropped using the DROP statement.

You can disable a function mapping with the mapping option DISABLE. Refer to the *SQL Reference* for more information on how to do this.

A UDF cannot be dropped if a view, trigger, table check constraint, or another UDF is dependent on it. Functions implicitly generated by the CREATE DISTINCT TYPE statement cannot be dropped. It is not possible to drop a function that is in either the SYSIBM schema or the SYSFUN schema.

Other objects can be dependent on a function or function template. All such dependencies, including function mappings, must be removed before the function can be dropped, with the exception of packages which are marked inoperative. Such a package is not implicitly rebound. It must either be rebound using the BIND or REBIND commands or it must be prepared by use of the PREP command. Refer to the *Command Reference* manual for more information on these commands. Dropping a UDF invalidates any packages or cached dynamic SQL statements that used it.

Dropping a function mapping marks a package as invalid. Automatic rebind will take place and the optimizer will attempt to use the local function. In the case where the local function is a template, the implicit rebind will fail.

(For more information, see "Statement Dependencies When Changing Objects" on page 215.)

Dropping a User-Defined Type (UDT) or Type Mapping

You can drop a user-defined type (UDT) or type mapping using the DROP statement. You cannot drop a UDT if it is used:

- In a column definition for an existing table or view (distinct types)
- As the type of an existing typed table or typed view (structured type)
- As the supertype of another structured type

You cannot drop a default type mapping; you can only override it by creating another type mapping.

The database manager attempts to drop all functions that are dependent on this distinct type. If the UDF cannot be dropped, the UDT cannot be dropped. A UDF cannot be dropped if a view, trigger, table check constraint, or another UDF is dependent on it. Dropping a UDT invalidates any packages or cached dynamic SQL statements that used it.

If you have created a transform for a UDT, and you are planning to drop the UDT, you should consider if it is necessary to drop the transform. This is done through the DROP TRANSFORM statement. Refer to the *SQL Reference* for details on this statement. Note that only transforms defined by you or other application developers can be dropped; built-in transforms and their associated group definitions cannot be dropped.

For more information about the user-defined types, refer to the *SQL Reference* and *Application Development Guide* manuals.

Altering or Dropping a View

The ALTER VIEW statement modifies an existing view by altering a reference type column to add a scope. Any other changes you make to a view require that you drop and then re-create the view.

When altering the view, the scope must be added to an existing reference type column that does not already have a scope defined. Further, the column must not be inherited from a superview.

The data type of the column-name in the ALTER VIEW statement must be REF (type of the typed table name or typed view name).

Other database objects such as tables and indexes are not affected although packages and cached dynamic statements are marked invalid. See "Statement Dependencies When Changing Objects" on page 215 for more information.

Refer to the *SQL Reference* for additional information on the ALTER VIEW statement.

To alter a view using the Control Center:

1. Expand the object tree until you see the **Views** folder.
2. Right-click on the view you want to modify, and select **Alter** from the pop-up menu.
3. In the **Alter view** window, enter or modify a comment, and click **Ok**.

To alter a view using the command line, enter:

```
ALTER VIEW <view_name> ALTER <column name>
   ADD SCOPE <typed table or view name>
```

To drop a view using the Control Center:

1. Expand the object tree until you see the **Views** folder.
2. Right-click on the view you want to drop, and select **Drop** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To drop a view using the command line, enter:

```
DROP VIEW <view_name>
```

The following example shows how to drop the EMP_VIEW:

```
DROP VIEW EMP_VIEW
```

Any views that are dependent on the view being dropped will be made inoperative. (See "Recovering Inoperative Views" for more information.)

As in the case of a table hierarchy, it is possible to drop an entire view hierarchy in one statement by naming the root view of the hierarchy, as in the following example:

```
DROP VIEW HIERARCHY VPerson
```

For more information on dropping and creating views, refer to the *SQL Reference* manual.

Recovering Inoperative Views

Views can become *inoperative*:

- As a result of a revoked privilege on an underlying table
- If a table, alias, or function is dropped
- If the superview becomes inoperative
- When the views they are dependent on are dropped.

The following steps can help you recover an inoperative view:

1. Determine the SQL statement that was initially used to create the view. You can obtain this information from the TEXT column of the SYSCAT.VIEW catalog view.
2. Re-create the view by using the CREATE VIEW statement with the same view name and same definition.
3. Use the GRANT statement to re-grant all privileges that were previously granted on the view. (Note that all privileges granted on the inoperative view are revoked.)

If you do not want to recover an inoperative view, you can explicitly drop it with the DROP VIEW statement, or you can create a new view with the same name but a different definition.

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

Dropping a Summary Table

You cannot alter a summary table, but you can drop it.

All indexes, primary keys, foreign keys, and check constraints referencing the table are dropped. All views and triggers that reference the table are made

inoperative. All packages depending on any object dropped or marked inoperative will be invalidated. See “Statement Dependencies When Changing Objects” on page 215 for more information on package dependencies.

To drop a summary table using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the summary table you want to drop, and select **Drop** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To drop a summary table using the command line, enter:

```
DROP TABLE <table_name>
```

The following SQL statement drops the summary table XT:

```
DROP TABLE XT
```

Recovering Inoperative Summary Tables

Summary tables can become *inoperative* as a result of a revoked SELECT privilege on an underlying table.

The following steps can help you recover an inoperative summary table:

- Determine the SQL statement that was initially used to create the summary table. You can obtain this information from the TEXT column of the SYSCAT.VIEW catalog view.
- Re-create the summary table by using the CREATE SUMMARY TABLE statement with the same summary table name and same definition.
- Use the GRANT statement to re-grant all privileges that were previously granted on the summary table. (Note that all privileges granted on the inoperative summary table are revoked.)

If you do not want to recover an inoperative summary table, you can explicitly drop it with the DROP TABLE statement, or you can create a new summary table with the same name but a different definition.

An inoperative summary table only has entries in the SYSCAT.TABLES and SYSCAT.VIEWS catalog views; all entries in the SYSCAT.VIEWDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS and SYSCAT.COLAUTH catalog views are removed.

Dropping a Wrapper

The DROP statement can remove a wrapper from the database. The following example shows how to drop the DRDA wrapper:

```
DROP WRAPPER DRDA
```

All server definitions, user-defined function mappings, and user-defined data type mappings that are dependent on the wrapper are dropped. All user-defined mappings, nicknames, user-defined type mappings, and user mappings that are dependent on the dropped server definitions are also dropped. Any index specifications dependent on the dropped nicknames are dropped, and any views dependent on these nicknames are marked inoperative. All packages dependent on the dropped objects and inoperative views are invalidated.

You must hold one of the SYSADM or DBADM authorities to DROP wrappers.

Refer to the *SQL Reference* for more information on dropping wrappers.

Altering or Dropping a Server

The ALTER SERVER statement modifies an existing server definition in the federated database catalog. Use this statement to:

- Modify the definition of a specific data source.
- Modify the definition of multiple data sources of a specific type or version.
- Make changes in the configuration of a specific data source. For example, if the DBMS identified by a specific server is migrated to a new workstation with a faster processor, you should update the cpu_ratio server option.

You cannot use this statement to modify the *dbname* or *node* server options.

The following example shows how to alter the ORA1 server:

```
ALTER SERVER ORA1 OPTIONS (SET CPU_RATIO '5.0')
```

Servers can be dropped from the federated database. The following example shows how to drop the ORALOC01 Server:

```
DROP SERVER ORALOC01
```

All nicknames for tables and views residing at the data source are dropped. Any index specifications dependent on these nicknames are dropped. Any user-defined function mappings, user-defined type mappings, and user mappings that are dependent on the dropped server definition are also dropped. All packages dependent on the dropped server definition, function mappings, nicknames, and index specifications are invalidated.

You must hold one of the SYSADM or DBADM authorities to ALTER or DROP servers.

For more information on dropping and altering servers, refer to the *SQL Reference*.

Altering or Dropping a Nickname

The ALTER NICKNAME statement is used to update locally stored information about a data source table or view. You could use this statement, for example, to change the local name for a column or to map a column data type to a different data type. You can also use this statement to add column options. For more information on ALTER NICKNAME syntax, see the *SQL Reference*.

When a nickname is dropped, views created on that nickname are marked as inoperative. You cannot alter nickname column names or data types when the nickname is referenced in a view.

You must hold one of the SYSADM or DBADM authorities, or, you must have either the CONTROL or ALL database privilege on the nickname, the ALTERIN (for the current schema) schema privilege, or be the nickname definer at the federated database to use this statement.

Altering a Nickname Column and Dropping a Nickname

The following example shows how to alter the nickname TESTNN, changing the local name of a column from COL1 to NEWCOL:

```
ALTER NICKNAME TESTNN ALTER COLUMN COL1 LOCAL NAME NEWCOL
```

The following example shows how to drop the nickname TESTNN:

```
DROP NICKNAME TESTNN
```

Altering Nickname Column Options

You specify column information in the form of values that you assign to parameters called *column options.* You can specify any of these values in either upper- or lowercase. The table below describes the values and provides additional information.

Table 3. Column Options and Their Settings

Option	Valid Settings		Default Setting
numeric_string	'Y'	Yes, this column contains only strings of numeric data. IMPORTANT: If this column contains only numeric strings followed by trailing blanks, it is inadvisable to specify 'Y'.	'N'
	'N'	No, this column is not limited to strings of numeric data.	
	By setting numeric_string to 'Y' for a column, you are informing the optimizer that this column contains no blanks that could interfere with sorting of the column's data. This option is helpful when the collating sequence of a data source is different from DB2. Columns marked with this option will not be excluded from local (data source) evaluation because of a different collating sequence.		
varchar_no_trailing_blanks	Indicates whether trailing blanks are absent from a specific VARCHAR column:		'N'
	'Y'	Yes, trailing blanks are absent from this VARCHAR column.	
	'N'	No, trailing blanks are not absent from this VARCHAR column.	
	If data source VARCHAR columns contain no padded blanks, then the optimizer's strategy for accessing them depends in part on whether they contain trailing blanks. By default, the optimizer "assumes" that they actually do contain trailing blanks. On this assumption, it develops an access strategy that involves modifying queries so that the values returned from these columns are the ones that the user expects. If, however, a VARCHAR column has no trailing blanks, and you let the optimizer know this, it can develop a more efficient access strategy. To tell the optimizer that a specific column has no trailing blanks, specify that column in the ALTER NICKNAME statement (for syntax, see the *SQL Reference*).		

Dropping an Index, Index Extension, or an Index Specification

You cannot change any clause of an index definition, index extension, or index specification; you must drop the index or index extension and create it again. (Dropping an index or an index specification does not cause any other objects to be dropped but may cause some packages to be invalidated.)

To drop an index, index extension, or an index specification using the Control Center:

1. Expand the object tree until you see the **Indexes** folder.
2. Right-click on the index you want to drop, and select **Drop** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To drop an index, index extension, or an index specification using the command line, enter:

```
DROP INDEX <index_name>
```

The following SQL statement drops the index called PH:

```
DROP INDEX PH
```

The following SQL statement drops the index extension called IX_MAP:

```
DROP INDEX EXTENSION ix_map RESTRICT
```

The name of the index extension must identify an index extension described in the catalog. The RESTRICT clause enforces the rule that no index can be defined that depends on the index extension definition. If an underlying index depends on this index extension, then the drop fails.

A primary key or unique key index (unless it is an index specification) cannot be explicitly dropped. You must use one of the following methods to drop it:

- If the primary index or unique constraint was created automatically for the primary key or unique key, dropping the primary key or unique key will cause the index to be dropped. Dropping is done through the ALTER TABLE statement.
- If the primary index or the unique constraint was user-defined, the primary key or unique key must be dropped first, through the ALTER TABLE statement. After the primary key or unique key is dropped, the index is no longer considered the primary index or unique index, and it can be explicitly dropped.

Any packages and cached dynamic SQL statements that depend on the dropped indexes are marked invalid. See "Statement Dependencies When Changing Objects" for more information. The application program is not affected by changes resulting from adding or dropping indexes.

Statement Dependencies When Changing Objects

Statement dependencies include package and cached dynamic SQL statements. A *package* is a database object that contains the information needed by the database manager to access data in the most efficient way for a particular application program. *Binding* is the process that creates the package the

database manager needs in order to access the database when the application is executed. The *Application Development Guide* discusses how to create packages in detail.

Packages and cached dynamic SQL statements can be dependent on many types of objects. Refer to the *SQL Reference* for a complete list of those objects.

These objects could be explicitly referenced, for example, a table or user-defined function that is involved in an SQL SELECT statement. The objects could also be implicitly referenced, for example, a dependent table that needs to be checked to ensure that referential constraints are not violated when a row in a parent table is deleted. Packages are also dependent on the privileges which have been granted to the package creator.

If a package or cached dynamic SQL statement depends on an object and that object is dropped, the package or cached dynamic SQL statement is placed in an ″invalid″ state. If a package depends on a user-defined function and that function is dropped, the package is placed in an ″inoperative″ state.

A cached dynamic SQL statement that is in an invalid state is automatically re-optimized on its next use. If an object required by the statement has been dropped, execution of the dynamic SQL statement may fail with an error message.

A package that is in an invalid state is implicitly rebound on its next use. Such a package can also be explicitly rebound. If a package was marked invalid because a trigger was dropped, the rebound package no longer invokes the trigger.

A package that is in an inoperative state must be explicitly rebound before it can be used. Refer to the *Application Development Guide* for more information about binding and rebinding packages.

Federated database objects have similar dependencies. For example, dropping a server invalidates any packages or cached dynamic SQL referencing nicknames associated with that server.

In some cases, it is not possible to rebind the package. For example, if a table has been dropped and not re-created, the package cannot be rebound. In this case, you need to either re-create the object or change the application so it does not use the dropped object.

In many other cases, for example if one of the constraints was dropped, it is possible to rebind the package.

The following system catalog views help you to determine the state of a package and the package's dependencies:

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

For more information about object dependencies, refer to the DROP statement in the *SQL Reference* manual.

Part 3. Database Security

Chapter 5. Controlling Database Access

One of the most important responsibilities of the database administrator and the system administrator is database security. Securing your database involves several activities:

- Preventing accidental loss of data or data integrity through equipment or system malfunction.
- Preventing unauthorized access to valuable data. You must ensure that sensitive information is not accessed by those without a "need to know".
- Preventing unauthorized persons from committing mischief through malicious deletion or tampering with data.
- Monitoring access of data by users which is discussed in "Chapter 6. Auditing DB2 Activities" on page 273.

The following topics are discussed:

- "Selecting User IDs and Groups for Your Installation"
- "Selecting an Authentication Method for Your Server" on page 225
- "Authentication Considerations for Remote Clients" on page 230
- "Partitioned Database Considerations" on page 230
- "Using DCE Security Services to Authenticate Users" on page 231
- "Federated Database Authentication Processing" on page 237
- "Privileges, Authorities, and Authorization" on page 242
- "Controlling Access to Database Objects" on page 255
- "Tasks and Required Authorizations" on page 266
- "Using the System Catalog" on page 267.

Planning for Security: Start by defining your objectives for a database access control plan, and specifying who shall have access to what and under what circumstances. Your plan should also describe how to meet these objectives by using database functions, functions of other programs, and administrative procedures.

Selecting User IDs and Groups for Your Installation

Security issues are important to the DB2 Administrator from the moment the product is installed. The respective platform-specific *Quick Beginnings* books present all of the information required to plan for, install, and configure DB2.

The steps to completing the installation of DB2 require a user name, a group name, and a password. During the installation, the administrator has default

values for each of these requirements. Once the defaults have been used during the installation of DB2, the administrator is strongly recommended to create new user names, group names, and passwords before creating the instances where the databases will reside. Using new user names, group names, and passwords will minimize the risk of a user other than the administrator learning of the defaults and using them in an improper fashion within instances and databases.

Passwords are very important when authenticating users. If no authentication requirements are set at the operating system level and the database is using the operating system to authenticate users, then users will be allowed to connect. For example on a UNIX operating system, undefined passwords are treated as NULL. And any user without a defined password will be treated as if they have a NULL password. From the operating system's perspective, this is a match and the user is validated and able to connect to the database. You should require passwords at the operating system level if you want the operating system to do the authentication of users for your database.

Another security recommendation following the installation of DB2 is the changing of the default privileges granted to users. During the installation process, System Administration (SYSADM) privileges are granted by default to the following users on each operating system:

OS/2 A valid DB2 user ID which belongs to the User Profile Management (UPM) Administrator or Local Administrator group.

Windows 95 or Windows 98 Any Windows 95 or Windows 98 user.

Windows NT or Windows 2000 A valid DB2 username which belongs to the Administrators group.

UNIX A valid DB2 username which belongs to the primary group of the instance owner's user ID.

SYSADM privileges are the most powerful set of privileges available within DB2. (Privileges are discussed later in this chapter.) As a result, you may not want all of these users to have SYSADM privileges by default. DB2 provides the administrator with the ability to grant and revoke privileges to groups and individual user IDs.

The platform-specific information to create and assign groups and user IDs is found in the various *Quick Beginnings* books. By updating the database manager configuration parameter SYSADM_GROUP, the administrator can control which group is defined as the System Administrative group with System Administrator privileges. You must follow the guidelines below to

complete the security requirements for both DB2 installation and the subsequent instance and database creation.

Any group defined as the System Administration group (by updating SYSADM_GROUP) must exist. The name of this group should allow for easy identification as the group created for instance owners. User IDs and groups that belong to this group have system administrator authority for their respective instances.

You should consider creating an instance owner user ID that is easily recognized as being associated with a particular instance. This user ID should have as one of its groups, the name of the SYSADM group created above. Another recommendation is to only use this instance owner user ID as a member of the instance owner group and not to use it in any other group. This should control the proliferation of user IDs and groups that could modify the instance environment.

The created user ID should always be associated with a password to allow for authentication before entry into the data and databases within the instance. The recommendation when creating a password is to follow your organization's password naming guidelines.

Windows NT Platform Considerations

When working in the Enterprise – Extended Edition for Windows NT, System Administration (SYSADM) authority is granted to any valid DB2 user account which belongs to the local Administrators group on the machine where the account is defined.

For example, if a user logs on to a domain account and tries to access a DB2 database, DB2 goes to a Domain Controller to enumerate groups (including the Administrator's group). You can change this behavior in either of two ways:

1. Set the registry variable DB2_GRP_LOOKUP = local and add the domain accounts (or global groups) to the local Administrators group.
2. Update the database manager configuration file to specify a new group. If you want that group enumerated on the local machine, then you must also set the DB2_GRP_LOOKUP registry variable.

By default in a Windows NT domain environment, only domain users that belong to the Administrators group at the Primary Domain Controller (PDC) have SYSADM authority on an instance. Since DB2 always performs authorization at the machine where the account is defined, adding a domain user to the local Administrators group on the server does not grant the domain user SYSADM authority to the group.

To avoid adding a domain user to the Administrators group at the PDC, you should create a global group and add the users (both domain and local) that you want to grant SYSADM authority. To do this, enter the following commands:

```
DB2STOP
DB2 UPDATE DBM CFG USING SYSADM_GROUP global_group
DB2START
```

UNIX Platform Considerations

On UNIX-based platforms, a group for fenced User Defined Functions (UDFs) and stored procedures must be created, and any user IDs that use fenced UDFs or stored procedures must be a member of this group. As with the SYSADM group, the name of the fenced UDFs or stored procedures group should allow for easy identification. User IDs that belong to the fenced UDFs or stored procedures group have whatever authority and privileges that are associated with the group as their default.

For security reasons, we recommend you do not use the instance name as the Fenced ID. However, if you are not planning to use fenced UDFs or stored procedures, you can set the Fenced ID to the instance name instead of creating another user ID.

The recommendation is to create a user ID that will be recognized as being associated with this group. The user for fenced UDFs and stored procedures is specified as a parameter of the instance creation script (db2icrt ... -u <FencedID>). This is not required if you install the DB2 Clients or the DB2 Software Developer's Kit.

General Rules

There are rules for the naming of all objects and users. Some of these rules are specific to the platform you are working on. For example, there is a rule regarding the use of upper and lower case letters in a name.

- On UNIX platforms, names must be in lower case.
- On OS/2, names must be in upper case.
- On Windows platforms, names can be in upper, lower, and mixed-case.

See "Appendix A. Naming Rules" on page 313 for DB2 naming rules.

The db2icrt command creates the main SQL library (sqllib) directory under the home directory of the instance owner.

Selecting an Authentication Method for Your Server

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. Refer to "Configuring DB2" in *Administration Guide: Performance* for more information on the *authentication* database manager configuration parameter. There is one authentication type per instance, which covers access to that database server and all the databases under its control.

If you intend to access data sources from a federated database, you must consider data source authentication processing and definitions for federated authentication types. See "Federated Database Authentication Processing" on page 237 for more information.

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.

If the remote instance has SERVER authentication, there are two ways that authentication can take place:

- The user ID and password are provided by the user.
- The user ID and password are retrieved by DB2 and then passed to the server for validation. (The user is already logged on to the local machine or to the domain.)

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 DCS or SERVER, the client is authenticated by passing the user ID and password to the server. If the client authentication is DCS_ENCRYPT or SERVER_ENCRYPT, the client is authenticated by passing a 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.

If the user performs a local or client login, the user is known only to that local client workstation.

If the remote instance has CLIENT authentication, two other parameters determine the final authentication type: *trust_allclnts* and *trust_clntauth*.

CLIENT level security for TRUSTED clients only:

Trusted clients are clients that have a reliable, local security system. Specifically, all clients are trusted clients except for Windows 95 and Windows 98 operating systems.

When the authentication type of CLIENT has been selected, an additional option may be selected to protect against clients whose operating environment has no inherent security.

To protect against unsecured clients, the administrator can select Trusted Client Authentication by setting the *trust_allclnts* parameter to NO. This implies that all trusted platforms can authenticate the user on behalf of the server. Untrusted clients are authenticated on the Server and must provide a user ID and password. You use the *trust_allclnts* configuration parameter to indicate whether you are trusting clients. The default for this parameter is YES.

Note: It is possible to trust all clients (*trust_allclnts* is YES) yet have some of those clients as those who do not have a native safe security system for authentication.

You may also want to complete authentication at the server even for trusted clients. To indicate where to validate trusted clients, you use the *trust_clntauth* configuration parameter. The default for this parameter is CLIENT. Refer to "Configuring DB2" in *Administration Guide: Performance* for more information on this parameter.

Note: For trusted clients only, if no user ID or password is explicitly provided when attempting to CONNECT or ATTACH, then validation of the user takes place at the client. The *trust_clntauth* parameter is only used to determine where to validate the information provided on the USER/USING clauses.

To protect against all clients except DRDA clients from DB2 for MVS and OS/390, DB2 for VM and VSE, and DB2 for OS/400, set the *trust_allclnts* parameter to DRDAONLY. Only these clients can be trusted to perform client-side authentication. All other clients must provide a user ID and password to be authenticated by the server.

The *trust_clntauth* parameter is used to determine where the above clients are authenticated: if *trust_clntauth* is "client", authentication takes place at the client. If *trust_clntauth* is "server", authentication takes place at the client when no password is provided and at the server when a password is provided.

Table 4. Authentication Modes using TRUST_ALLCLNTS and TRUST_CLNTAUTH Parameter Combinations.

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	Untrusted non–DRDA Client Authen-tication no password	Untrusted non–DRDA Client Authen-tication with password	Trusted non–DRDA Client Authen-tication no password	Trusted non–DRDA Client Authen-tication with password	DRDA Client Authen-tication no password	DRDA Client Authen-tication with password
YES	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT
YES	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
NO	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
NO	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

DCS Primarily used to catalog a database accessed using DB2 Connect. (Refer to the *DB2 Connect User's Guide* section on Security for more details on this topic.) When it is used to specify the authentication type for an instance in the database manager configuration file, it means the same as authentication **SERVER**, unless the server is being accessed via the Distributed Relational Database Architecture (DRDA) Application Server (AS) architecture using the Advanced Program-To-Program Communications (APPC) protocol. In this case, using **DCS** indicates that authentication will occur at the server, but

only in the APPC layer. Further authentication will not occur in the DB2 code. This value is only supported when the APPC SECURITY parameter for the connection is specified as SAME or PROGRAM.

DCS_ENCRYPT
Specifies that DB2 Connect 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 DCS or SERVER, the client is authenticated by passing the user ID and password to DB2 Connect. If the client authentication is DCS_ENCRYPT or SERVER_ENCRYPT, the client is authenticated by passing a user ID and encrypted password.

If DCS_ENCRYPT is specified at the client and DCS is specified at the server, an error is returned because of the mismatch in the authentication levels.

DCE Specifies that the user is authenticated using DCE Security Services. For more information on DCE Security, see "Using DCE Security Services to Authenticate Users" on page 231.

DCE_SERVER_ENCRYPT
Specifies that the server accepts DCE authentication or encrypted SERVER authentication schemes. If the client authentication is DCE or not specified, the client is authenticated using DCE Security Services. For more information on DCE Security, see "Using DCE Security Services to Authenticate Users" on page 231.

If the client authentication is SERVER or DCS, the client is authenticated by passing the user ID and password to the server. If the client authentication is SERVER_ENCRYPT or DCS_ENCRYPT, the client is authenticated by passing a user ID and encrypted password. The authentication type of the client cannot be specified as DCE_SERVER_ENCRYPT. If the authentication type of an instance is specified as DCE_SERVER_ENCRYPT, all local applications will use DCE as the authentication scheme. This also applies for any utility commands that do not require a database connection or an instance attachment.

In addition to allowing a mix of DCE and SERVER_ENCRYPT authentication types, the DCE_SERVER_ENCRYPT authentication type also alleviates one of the limitations when using groups within DCE. When the authentication type is set to DCE_SERVER_ENCRYPT, the assumption is that the group list being requested other than at authentication time, come from the base operating system and not from DCE. You, as the administrator, can then set up a user on the server to match the short DCE name in order to provide group list support outside that which is supported at authentication time.

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.

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, 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.

Notes:

1. The type of authentication you choose is important only if you have remote database clients accessing the database or when you are using federated database functionality. Most users accessing the database through local clients are always authenticated on the same machine as the database. An exception can exist when DCE Security Services are used. For information about supporting and using remote clients, refer to your *Quick Beginnings* manual.
2. Do not inadvertently lock yourself out of your instance when you are changing the authentication information, since access to the configuration file itself is protected by information in the configuration file. The following database manager configuration file parameters control access to the instance:
   - AUTHENTICATION *
   - SYSADM_GROUP *
   - TRUST_ALLCLNTS
   - TRUST_CLNTAUTH
   - SYSCTRL_GROUP
   - SYSMAINT_GROUP

   * Indicates the two most important parameters, and those most likely to cause a problem.

There are some things that can be done to ensure this does not happen: If you do accidentally lock yourself out of the DB2 system, you have a fail-safe option available on all platforms that will allow you to override the usual DB2 security checks to update the database manager configuration file using a highly privileged local operating system security user. This user **always** has the privilege to update the database manager configuration file and thereby correct the problem. However, this security bypass is restricted to a local update of the database manager configuration file. You cannot use a fail-safe user remotely or for any other DB2 command. This special user is identified as follows:

- UNIX platforms: the instance owner
- NT platform: someone belonging to the local "administrators" group
- OS/2 platform: a UPM administrator
- Other platforms: there is no local security on the other platforms, so all users pass local security checks anyway

3. See "Appendix E. How DB2 for Windows NT Works with Windows NT Security" on page 361 for additional information on Windows NT Security.

Authentication Considerations for Remote Clients

When cataloging a database for remote access, the authentication type may be specified in the database directory entry.

For databases accessed using DB2 Connect: If a value is not specified, SERVER authentication is assumed.

For databases accessed remotely but not using DB2 Connect: The authentication type is not required. However, if it is not specified the client must first contact the server to obtain the value before beginning the authentication flow. If specified, authentication can begin immediately provided the value specified matches that at the server. If a mismatch is detected then DB2 attempts to recover which may result in more flows to reconcile the difference or in an error if DB2 cannot recover. In the case of a mismatch, the value at the server is assumed to be correct.

Partitioned Database Considerations

In a partitioned database, each partition of the database must have the same set of users and groups defined. If the definitions are not the same, the user may be authorized to do different things on different partitions. Consistency across all partitions is recommended.

Using DCE Security Services to Authenticate Users

When considering security for your distributed database environment, Distributed Computing Environment (DCE) Security Services are a good option because DCE provides:

- Centralized administration of users and passwords.
- No transmission of clear text passwords and user IDs.
- A single sign-on for users.

DB2 supports DCE default login contexts, connection login contexts, and delegated contexts. A *default login context* is established when a user does a `dce_login` on a client. Subsequent DB2 commands have access to this context and may perform user authentication without further user intervention (that is, no requirement for a user ID or password). A *connection login context* is established for a DB2 session using the user ID and password provided on CONNECT or ATTACH using the USER/USING clause. Finally, a *delegated login context* occurs when a DB2 client is used as part of a DCE server application. The DCE server application (that is also a DB2 client), receives requests from a DCE client application, from which point the original identity of the user originates. Provided the DCE client and DCE server are correctly configured to allow the DCE server to be a delegate for the DCE client, DB2 will obtain the delegated token and forward this to the DB2 server. This allows the DB2 server to use the original identity of the DCE client, rather than using the identity of the DCE server, to process requests. Information on how to establish a delegated login context can be obtained from the DCE documentation for your platform.

Note: There are several vendor products that support DCE. To ensure that DB2 UDB for Windows NT can work with IBM's DCE product in the area of security services, two new DLLs have been provided: `db2dces.ibm` and `db2dcec.ibm`. (These DLL files are only appropriate for Windows NT.) If you purchase and use IBM's DCE product for security services, these two files must be copied to `db2dces.dll` and `db2dcec.dll` respectively. If you are considering another vendor's DCE product, you should contact the vendor service organization and the DB2 UDB service organization to discuss whether the vendor's DCE implementation for security services will work with DB2 UDB.

How to Set up a DB2 User for DCE

Users must be registered in the Distributed Computing Environment (DCE) Registry and have correct attributes before being used with DB2. See the appropriate platform-specific DCE documentation for information on how to create a DCE principal.

Each DB2 user wishing to use a DCE-authenticated server must have a DCE principal and account defined in the DCE Registry with the client flag

enabled. This principal must also have an entry in its Extended Registry Attributes (ERA) section showing what authorization name will be used for this principal when it connects to a particular DCE authenticated server.

You may also wish to have user principals be members of groups in order to use group privileges in the database. Similar information in the group ERA maps the group name to a DB2 authorization name. The authorization name is a secondary authorization name but the same restrictions apply. Please refer to your DCE documentation for additional information on how to create groups and add members.

The information in the ERA maps a user's DCE principal or group name to a DB2 authorization name for a particular server's DCE principal name. To use an ERA, an ERA schema indicating the format of this attribute must be defined. This needs to be done once per DCE cell and is accomplished by completing the following steps:

1. Login to DCE as a valid DCE administrator
2. Invoke `dcecp` and enter the following at the prompt:

```
> xattrschema create /.:/sec/xattrschema/db2map \
> -aclmgr {{principal r m r m } {group r m r m }} \
> -annotation {Schema entry for DB2 database access} \
> -encoding stringarray \
> -multivalued no \
> -uuid 1cbe84ca-9df3-11cf-84cd-02608c2cd17b
```

This creates the Extended Registry Attribute `db2map`.

To view this mapping, issue the following command at the dcecp prompt:

```
> xattrschema show /.:/sec/xattrschema/db2map
```

You will see the following:

```
{axlmgr
 {{principal {{query r} {update m} {test r} {delete m}}}
  {group     {{query r} {update m} {test r} {delete m}}}}}
{annotation {Schema entry for DB2 database access}}
{applydefs no}
{intercell rejects}
{multivalued no}
{reserved no}
{scope {}}
{trigbind {}}
{trigtype none}
{unique no}
{uuid 1cbe84ca-9df3-11cf-84cd-02608c2cd17b}
```

Note: Restrictions on the contents of the authorization name recorded in the ERA are not enforced by DCE. If a DCE principal or group is given an invalid authorization name, an error results when an attempt is made

by DB2 to authenticate that user. (Recall that authentication may occur at CONNECT, ATTACH, DB2START, or any other operation where authentication is required.) It is also highly recommended that you ensure the assignment of authorization names to DCE principals is one-to-one and unique. DCE does not check these conditions.

If a DB2 client is to access a DB2 UDB server, once they are registered as DCE principals, the ERA information must be added to provide the mapping from the principal name to the authorization name. This must be done once for each user or group; and, is accomplished by completing the following steps:

- Login to DCE as a valid DCE administrator
- Invoke `dcecp` and at the prompt enter the following:

```
> principal modify principal_name \
> -add {db2map map_1 map_2...map_n}
```

where `map_n` uses the following format:

```
DCE_server_principal,DB2_authid
```

where `DCE_server_principal` is a valid DCE principal name for a DB2 UDB server (or is the wildcard * which indicates this mapping is valid for any DB2 server not already specified in another `map_n` entry) and `DB2_authid` is a valid DB2 authorization name.

If a DCE group is to be used for a DCE principal, it must also have a mapping to a DB2 authid which has the proper authority such as SYSADM or SYSCTRL authority.

Please note that the authorization identifier (authid) specified in the DCE schema used to map a DCE principal name to a DB2 authid *must* be specified in uppercase. Use of a lowercase or mixed case authid will result in an error.

How to Setup a DB2 Server to Use DCE

Servers must be registered principals in the Distributed Computing Environment (DCE) Registry and have correct attributes before being used with DB2. See the appropriate platform-specific DCE documentation for information on how to create a DCE server principal.

The DCE Security client runtime code must be installed and accessible by the server instance.

Each DB2 server that wishes to use DCE as an authentication mechanism must register with DCE at the time of issuing DB2START. To avoid having to do this manually, DCE provides a method whereby a server maintains its own user ID and password (key) information in a special file called a *keytab* file. At DB2START, DB2 reads the database manager configuration file and obtains

the authentication type for the instance. If it finds the authentication type is DCE, DCE calls are made by the DB2 server to obtain the information from the keytab file. It is this information that is used to register the server with DCE. This registration allows the server to accept DCE tokens from DCE clients and to use them to authenticate these users.

The instance administrator must create the keytab file for the instance using DCE commands. Detailed information on how to create a keytab file is included in the DCE documentation for your platform. In that document, refer to the details associated with the keytab file and the commands *dcecp keytab* or *rgy_edit*. The DB2 keytab file must be named *keytab.db2* and must reside in the `security` subdirectory of the `sqllib` directory for the instance. (For Intel-based operating systems, the file must reside in the `security` subdirectory of the `INSTANCENAME` subdirectory of the `sqllib` directory. `INSTANCENAME` is the instance name of the instance you are working with.) It should contain only one entry for the server principal for the specified instance; anything else results in an error at DB2START time. On UNIX operating system platforms, this file must be protected with file permissions to only allow read/write for the instance owner.

Following is an example of the creation of the keytab file:

- Log in to DCE as a valid DCE user
- Invoke `rgy_edit`, and enter the following at the prompt:

```
> ktadd -p principal_name -pw principal_password \
> -f keytab.db2
```

To start DB2 using DCE authentication once the DCE configuration is complete, you must tell DB2 it is to use DCE authentication by updating the database manager configuration file with authentication type "DCE". This is done by issuing the following CLP command:

```
db2 update database manager configuration using authentication DCE
 sysadm_group DCE_group_name
```

Then perform a `dce_login` to a valid DB2 DCE user who has SYSADM authority and issue DB2START.

Note: Before starting DB2 using DCE authentication, ensure you have defined a DCE user principal to be used as your SYSADM for the instance so that you have a valid DCE user ID from which to start, stop, and administer the instance. Please see "How to Set up a DB2 User for DCE" on page 231 for instructions on how to do this.

In addition to these instructions, ensure the principal created is a member of the SYSADM_GROUP for the instance. By default, this group name is DB2ADMIN for DCE authentication when no group is

explicitly specified (that is, when the SYSADM_GROUP is null), but it can be updated before changing the authentication type for the instance to a group name (authorization name) of your choice. The DCE group that you select must have an ERA defined that maps it to the specified SYSADM_GROUP authorization name.

One of the functions of the DB2 Administration Server is to start DB2 instances. When AUTHENTICATION = DCE, the DCE principal used in the DB2 keytab file for the instance must have a valid DCE principal to DB2 authid mapping. This mapping is required for the DB2 Administration Server to start the DB2 instance. The valid mapping allows this ID to act as a client as well as a server.

How to Set up a DB2 Client Instance to Use DCE

A client-only instance may be established to use DCE authentication for local operations by updating the database manager configuration file and setting the authentication type to DCE. There is no requirement to have a keytab file for a client-only instance since there is no server that needs to register to DCE. In general, it is not recommended (or required) that a client-only DB2 instance use DCE authentication, but it is supported.

A client that wishes to access a remote database using DCE security requires access to the applicable DCE Security product. Optionally, the client may choose to catalog the authentication type for the target database in the database directory. If the client chooses to specify DCE authentication, the fully-qualified DCE server principal name must also be specified. If DCE authentication is not specified in the directory, the authentication and principal information is obtained from the server at CONNECT time.

DB2 Restrictions Using DCE Security

Using DCE authentication places some restrictions on certain SQL functions provided by DB2 and related to group support. The following restrictions exist when using DCE authentication:

- When using the GRANT or REVOKE statements, the keywords USER and GROUP **must** be specified to qualify the authorization name specified, otherwise an error is issued.
- When using the AUTHORIZATION clause of the CREATE SCHEMA statement, the group membership of the authorization name specified will **not** be considered in evaluating the authorizations required to perform the statements that follow this clause. This may result in an authorization failure during execution of the CREATE SCHEMA statement.
- When a package is rebound by a user other than the original binder of the package, the privileges of the original binder are reevaluated. In this case, group membership of the original binder are not considered when reevaluating privileges. This may result in an authorization failure during rebinding.

DCE authentication as performed by DB2 flows DCE Tickets obtained using the OSF DCE Generic Security Services Application Programming Interface (GSSAPI). As such, all authentication for DCE Security takes place at the database protocol layer. Certain communication mechanisms may provide additional communication layer security, which is not necessarily integrated with DCE. In cases where the communication layer authentication can be kept entirely independent of the database protocol layer authentication, no restrictions will be enforced. However, the criteria for both the database protocol layer and the communication layer authentication must be satisfied before a connection can be successfully established. In cases where the database protocol layer and the communication protocol layer authentication mechanisms interact, their use may be restricted if some combinations result in a security exposure.

DCE authentication may be used in conjunction with TCPIP SOCKS support; however, the two security mechanisms work independently of one another. This may mean that not only must the user provide a valid DCE login context, but must also be logged on to a local operating system user ID that meets the criteria of the SOCKS Server.

DCE authentication may be used in conjunction with NT Named Pipes; however, the two security mechanisms work independent of one another. Not only must the user provide a valid DCE login context, but he must also be logged on to the NT Domain to a user ID that meets the criteria for the NT Named Pipes support.

In order to address possible confusion where DCE principals and local operating system user IDs are both used for authentication, as in the above two examples, an integrated DCE logon can be used. In this case, when logging on to a system, the user is automatically logged into the appropriate DCE principal as well. See the DCE documentation for your platform for details on how to use this feature, if it is supported. Note that in using this approach, the same name is used for the DCE principal and the local operating system ID. This may mean that the same value that is contained in the DCE encrypted ticket also flows on the wire unencrypted in the communication layer.

DCE authentication can only be used with APPC communications when the SECURITY parameter is set to NONE. This is to avoid the possibility of sending an unencrypted principal and/or password in the communication layer, while using an encrypted DCE token for the same principal in the database protocol layer. DCE Security at the APPC layer is not supported by DB2 at this time.

Federated Database Authentication Processing

If you have installed the distributed join installation feature and set the database manager configuration variable *federated* to 'YES', your DB2 system is operating as a federated system. Database authentication settings in a federated system differ slightly from standard DB2 definitions. More importantly, in a federated system you must consider the authentication requirements of your data sources. In general, data sources (DB2, Oracle, DB2 for OS/390, and so on) are set up to require authentication. That means you must ensure that IDs and passwords (as required) can flow to data sources. DB2 provides several methods for supporting authentication at data sources, all of which are explained in this section.

Authentication Settings

SERVER
Specifies that clients connecting to DB2 provide a user ID and password to access DB2. In this case a user ID and password are available for transmission to data sources. You control what is actually passed to the data sources through server options and user mappings, but authentication information is available for transmission to the data source.

CLIENT
Specifies that authentication takes place on the database partition where the application is invoked using operating system security. No passwords are available for transmission directly to data sources. In this case, if a data source requires authentication, you must create one or more user mappings. You must also ensure that server options are set properly to transmit correct user ID and password information to the data source.

Exercise extreme caution when using CLIENT authentication. Consider this form of authentication only for secure networks. A user has SYSADM authority for the federated database when the following conditions are met:

- Authentication is set to CLIENT.
- The user has root status at the client.
- The user knows the SYSADM's authorization name.
- The user defines an authorization name on the client that is the same as the SYSADM's on DB2.

DCS Specifies that authentication takes place at a data source – not DB2. In this case, standard DB2 authentication processing is bypassed. User IDs and passwords are passed directly to data sources, depending on server option settings. Authentication takes place only at Oracle or DB2 Family data sources.

Exercise caution when authentication is set to DCS. Authentication is done at neither the client nor at DB2. Any user who knows the SYSADM authentication name can assume SYSADM authority for the federated server.

DCE If authentication is set to DCE, only a user ID is available for transmission to data sources. No password is available. If a data source requires authentication processing (user ID and password), you must define a user mapping that will transmit a password (and possibly a user ID) to the data source. If the data source trusts the DB2 connection, user mappings are not required because the ID received from the external security system can be passed to the data source.

Other DB2 authentication settings are possible, and one or more can result in the availability of a password at DB2 for transmission to data sources. If DB2 and client authentication settings result in the transmission of a password to DB2, that password is available for additional authentication processing at data sources. See Table 4 on page 227 for more information.

Passing User IDs and Passwords to Data Sources

There are four ways to control the transmission of authentication information to data sources: DB2 authentication settings, user mappings, server options, and APPC security settings:

Authentication Settings

The purpose of this section is to clarify how authentication settings influence global authentication processing in a federated system (the definitions for authentication settings are in "Authentication Settings" on page 237). For example, if DB2 authentication is set to SERVER or DCS, a user ID and password are required for a connection. Therefore, a user ID and password are available for transmission to data sources. If authentication is set to DCE or CLIENT, and authentication is not taking place at the DB2 system containing the federated database, only a user ID is available. If data source authentication processing requires a password (or perhaps a different user ID and a password), you must create a user mapping. If authentication is set to CLIENT, and the *trust_clntauth* parameter setting is SERVER, it is possible that a password is sent to DB2 and that it is available for transmission to data sources.

User Mappings

DB2 can send either the authorization name used to connect to DB2 or an authorization name defined at DB2. User mappings store authorization names defined at DB2. They are created with the CREATE USER MAPPING statement.

User mappings are flexible: you can map an ID to a new ID and password or just a password. You can use them to provide missing information or to change an ID and password to values accepted at the data source.

To create or alter a user mapping, you must hold one of the SYSADM or DBADM authorities, or your authentication ID must match the authorization name specified for the statement.

An example of a user mapping statement is:

```
CREATE USER MAPPING FOR "SHAWN" SERVER DB21 OPTIONS (REMOTE_AUTHID "SHAWNBCA",
REMOTE_PASSWORD "MAPLELEAF")
```

where a DB2 authentication ID (SHAWN) is mapped to the remote ID SHAWNBCA and remote password MAPLELEAF for a server named DB21.

If the only difference between the authorization name (or password) at DB2 and the authorization name (or password) at the data source is the case of the passed string, consider using server options to fold the case to the desired setting instead of creating new IDs and passwords. See "Server Options" for more information.

You must create a user mapping when your authentication setting is DCE and a data source requires authentication processing (a password is expected). DB2 will only pass the DCE user ID to data sources. A password must be mapped to that user ID and then sent to the data source.

Server Options

Server options can be used to provide overall authentication support. Use them to indicate if passwords are passed to data sources (typically yes) and whether user IDs and passwords need to be folded to uppercase or lowercase. Server options are set using the CREATE SERVER, ALTER SERVER, and SET SERVER OPTION statements.

Server options specific to authentication processing are discussed in the rest of this section. A more complete list of server options is in "Using Server Options to Help Define Data Sources and Facilitate Authentication Processing" on page 153.

Password Server Option: The default setting for password is 'Y' (passwords are sent to data sources). Leave or set this option to 'Y' for all cases where a data source will perform authentication and is not expecting an encrypted password.

DB2 can transmit encrypted passwords. Set the server option password to 'ENCRYPTION' if passwords should be sent in an encrypted form to DB2

Family data sources. It is recommended that you set password to 'ENCRYPTION' if your authentication setting at DB2 is DCS_ENCRYPT or SERVER_ENCRYPT.

A user ID is always sent to data sources.

ID and Password Folding Options: Authorization names and passwords, in some cases, might need to change. Different data sources can have different authorization name and password requirements (regarding the use of uppercase or lowercase) for IDs and passwords.

DB2 provides two server options that can help you resolve naming differences. The option names are **fold_id** and **fold_pw**, and their settings are:

'U' DB2 folds the authorization name or password to uppercase before sending it to the data source.

'N' DB2 does not fold the authorization name or password.

'L' DB2 folds the authorization name or password to lowercase before sending it to the data source.

null DB2 first sends the authorization name or password as uppercase; if that fails, DB2 folds it to lowercase and sends it again.

The null setting might seem attractive because it covers many possibilities. However, from a performance perspective, it is best to set these options so that only one attempt is made for connections. If both the fold_id and fold_pw options are set to null, it is possible that DB2 will make four attempts to send the authorization name and password:

1. Both authorization name and password in uppercase.
2. Authorization name in uppercase and password in lowercase.
3. Authorization name in lowercase and password in uppercase.
4. Both authorization name and password in lowercase.

APPC Security Settings

If you are connecting across APPC to a DRDA data source, that requires a user ID and password, or if your authentication setting is DCS and you are authenticating at a DRDA data source, ensure that your APPC security setting is PROGRAM for the connection between DB2 and that data source.

Federated Database Authentication Example

This section provides an overview of federated system authentication and authorization steps. See Figure 3 on page 241 for an overview of federated database authentication and authorization processing.

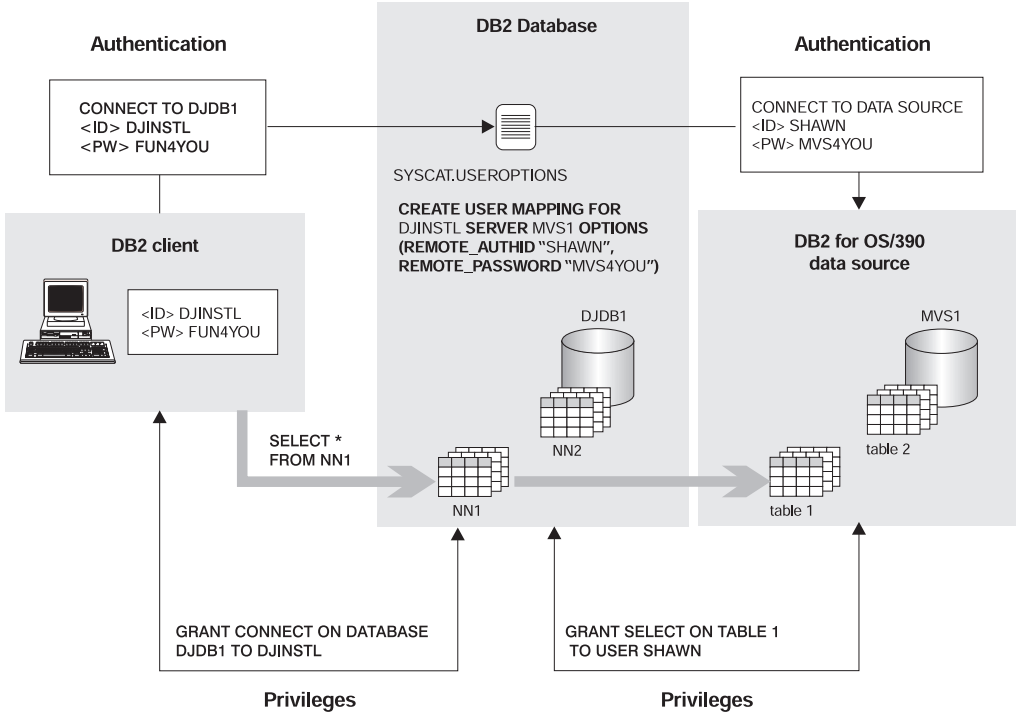


Figure 3. Federated Database Authentication and Authorization Processing

The task in this scenario is to enable the user DJINSTL to perform a UNION operation against two nicknames (NN1 and NN2). The nicknames represent two tables. One data source is a DB2 for OS/390 system where DJINSTL has a different user ID and password (see Figure 3) named MVS1. A user mapping will be required to access information at MVS1. The other data source is a DB2 system where DJINSTL's ID and password are the same. This data source, DB21, simply requires that the user ID and password are sent in uppercase.

DB2 authentication is set to SERVER. DJINSTL will access DB2 from a Windows NT client across a TCP/IP connection. The connection from DB2 to DB2 for OS/390 is also TCP/IP. The federated database name is DJDB1.

First ensure that DB2 is expecting a password and that a password is being sent. Also, ensure that the client and server authentication types match. Check the DB2 server authentication type by issuing the command:

```
GET DATABASE MANAGER CONFIGURATION
```

from the DB2 server. Check the client authentication type by issuing the command:

```
LIST DATABASE DIRECTORY
```

from the client. In both cases, ensure that authentication is set to SERVER. If the setting for the client is DCS or CLIENT, you can change it by using the UNCATALOG DATABASE and CATALOG DATABASE commands.

Next, ensure that passwords will be sent to the data sources. After connecting to the federated database DJDB1, issue the commands:

```
ALTER SERVER MVS1 OPTIONS (SET password 'Y')
ALTER SERVER DB21 OPTIONS (SET password 'Y')
```

Next, ensure that passwords are sent to the DB21 data source in the proper case:

```
ALTER SERVER DB21 OPTIONS (ADD fold_id 'U')
ALTER SERVER DB21 OPTIONS (ADD fold_pw 'U')
```

The next step is to grant privileges allowing the user DJINSTL to connect to the federated database DJDB1 and select nicknames:

```
GRANT CONNECT ON DATABASE DJDB1 TO DJINSTL;
```

Now, map DJINSTL's DB2 ID and password to the correct user ID and password for the MVS1 server:

```
CREATE USER MAPPING FOR "DJINSTL" SERVER MVS1 OPTIONS (REMOTE_AUTHID "SHAWN",
REMOTE_PASSWORD "MVS4YOU")
```

At this point, the DB2 user ID DJINSTL can send requests to data sources. Additional steps might be required to access data source objects referenced by nicknames (privileges are usually required for tables and views referenced by nicknames).

Privileges, Authorities, and Authorization

Privileges enable users to create or access database resources. *Authority levels* provide a method of grouping privileges and higher-level database manager maintenance and utility operations. Together, these act to control access to the database manager and its database objects. Users can access only those objects for which they have the appropriate *authorization*, that is, the required privilege or authority.

The following authorities exist:

- "System Administration Authority (SYSADM)" on page 244
- "System Control Authority (SYSCTRL)" on page 245
- "System Maintenance Authority (SYSMAINT)" on page 246
- "Database Administration Authority (DBADM)" on page 247
- "LOAD Authority" on page 247

The following types of privileges exist:

- "Database Privileges" on page 248
- "Schema Privileges" on page 249
- "Table Space Privileges" on page 250
- "Table and View Privileges" on page 251
- "Nickname Privileges" on page 253
- "Server Privileges" on page 254
- "Package Privileges" on page 254
- "Index Privileges" on page 255.

Figure 4 illustrates the relationship between authorities and their span of control (database, database manager).

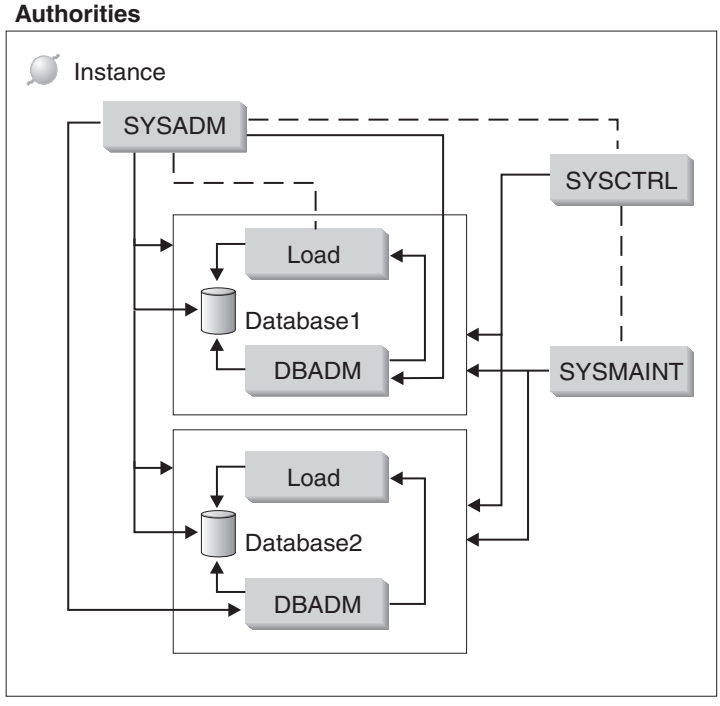


Figure 4. Hierarchy of Authorities

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

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

- LOAD authority (LOAD) gives LOAD utility or AutoLoader utility privileges to load data into tables.
- Ownership privilege (also called CONTROL privilege in some cases) gives full privileges for a specific object.
- Individual privileges may be granted to allow a user to carry out specific functions on specific objects.
- Implicit privileges may be granted to a user who has the privilege to execute a package. While users can run the application, they do not necessarily require explicit privileges on the data objects used within the package. For more information see "Allowing Indirect Privileges Through a Package" on page 259.

Users with administrative authority (SYSADM or DBADM) or ownership privileges (CONTROL) can grant and revoke privileges to and from others, using the GRANT and REVOKE statements. (See "Controlling Access to Database Objects" on page 255.) It is also possible to grant a table, view, or schema privilege to another user if that privilege is held WITH GRANT OPTION. However, the WITH GRANT OPTION does not allow the person granting the privilege to revoke the privilege once granted. You must have SYSADM authority, DBADM authority, or CONTROL privilege to revoke the privilege.

A user or group can be authorized for any combination of individual privileges or authorities. When a privilege is associated with a resource, that resource must exist. For example, a user cannot be given the SELECT privilege on a table unless that table has previously been created.

Note: Care must be taken when an authorization name is given authorities and privileges and there is no user created with that authorization name. At some later time, a user can be created with that authorization name and automatically receive all of the authorities and privileges associated with that authorization name.

Refer to the *Command Reference*, the *Administrative API Reference*, or the *SQL Reference* for information about what authorization is required for a particular command, API, or SQL statement.

System Administration Authority (SYSADM)

SYSADM authority is the highest level of administrative authority. Users with SYSADM authority can run utilities, issue database and database manager commands, and access the data in any table in any database within the database manager instance. It provides the ability to control all database objects in the instance, including databases, tables, views, indexes, packages, schemas, servers, aliases, data types, functions, procedures, triggers, table spaces, nodegroups, buffer pools, and event monitors.

SYSADM authority is assigned to the group specified by the *sysadm_group* configuration parameter (refer to "Configuring DB2" in *Administration Guide: Performance*). Membership in that group is controlled outside the database manager through the security facility used on your platform. Refer to the *Quick Beginnings* for information on how to use your system security facility to create, change, or delete SYSADM authorities.

Only a user with SYSADM authority can perform the following functions:

- Migrate a database
- Change the database manager configuration file (including specifying the groups having SYSCTRL or SYSMAINT authority)
- Grant DBADM authority.

In addition, a user with SYSADM authority can perform the functions of users with the following authorities:

- "System Control Authority (SYSCTRL)"
- "System Maintenance Authority (SYSMAINT)" on page 246
- "Database Administration Authority (DBADM)" on page 247

Note: When users with SYSADM authority create databases, they are automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSADM group, and if you want to also prevent them from accessing that database as a DBADM, you must explicitly revoke this DBADM authority.

System Control Authority (SYSCTRL)

SYSCTRL authority is the highest level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System control authority is designed for users administering a database manager instance containing sensitive data.

SYSCTRL authority is assigned to the group specified by the *sysctrl_group* configuration parameter (refer to "Configuring DB2" in *Administration Guide: Performance*). If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSCTRL authority or higher can do the following:

- Update a database, node, or distributed connection services (DCS) directory
- Force users off the system
- Create or drop a database
- Drop, create, or alter a table space

- Restore to a new database.

In addition, a user with SYSCTRL authority can perform the functions of users with "System Maintenance Authority (SYSMAINT)" authority.

Users with SYSCTRL authority also have the implicit privilege to connect to a database.

Note: When users with SYSCTRL authority create databases, they are automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSCTRL group, and if you want to also prevent them from accessing that database as a DBADM, you must explicitly revoke this DBADM authority.

System Maintenance Authority (SYSMAINT)

SYSMAINT authority is the second level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System maintenance authority is designed for users maintaining databases within a database manager instance that contains sensitive data.

SYSMAINT authority is assigned to the group specified by the *sysmaint_group* configuration parameter (refer to "Configuring DB2" in *Administration Guide: Performance*). If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSMAINT or higher system authority can do the following:

- Update database configuration files
- Back up a database or table space
- Restore to an existing database
- Perform roll forward recovery
- Start or stop an instance
- Restore a table space
- Run trace
- Take database system monitor snapshots of a database manager instance or its databases.

A user with SYSMAINT, DBADM, or higher authority can do the following:

- Query the state of a table space
- Update log history files

- Quiesce a table space
- Reorganize a table
- Collect catalog statistics using the RUNSTATS utility.

Users with SYSMAINT authority also have the implicit privilege to connect to a database.

Database Administration Authority (DBADM)

DBADM authority is the second highest level of administrative authority. It applies only to a specific database, and allows the user to run certain utilities, issue database commands, and access the data in any table in the database. When DBADM authority is granted, BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA privileges are granted as well. Only a user with SYSADM authority can grant or revoke DBADM authority. Users with DBADM authority can grant privileges on the database to others and can revoke any privilege from any user regardless of who granted it.

Only a user with DBADM or higher authority can do the following:

- Read log files
- Create, activate, and drop event monitors.

A user with DBADM, SYSMAINT, or higher authority can do the following:

- Query the state of a table space
- Update log history files
- Quiesce a table space.
- Reorganize a table
- Collect catalog statistics using the RUNSTATS utility.

Note: A DBADM can only perform the above functions on the database for which DBADM authority is held.

LOAD Authority

Users having LOAD authority at the database level, as well as INSERT privilege on a table, can use the LOAD command or the AutoLoader utility 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 load operation is 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, 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.

Database Privileges

Figure 5 shows the database privileges.

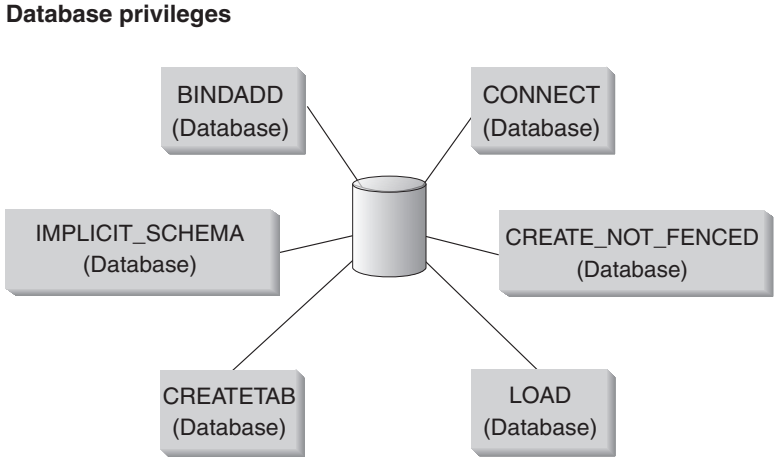


Figure 5. Database Privileges

Database privileges involve actions on a database as a whole:

- CONNECT allows a user to access the database
- BINDADD allows a user to create new packages in the database
- CREATETAB allows a user to create new tables in the database
- CREATE_NOT_FENCED allows a user to create a user-defined function (UDF) or procedure that is "not fenced". UDFs or procedures that are "not fenced" must be extremely well tested because the database manager does not protect its storage or control blocks from these UDFs or procedures. (As a result, a poorly written and tested UDF or procedure that is allowed to run "not fenced" can cause serious problems for your system.) (Refer to the *Application Development Guide* or the *SQL Reference* for more information.)
- IMPLICIT_SCHEMA allows any user to create a schema implicitly by creating an object using a CREATE statement with a schema name that does not already exist. SYSIBM becomes the owner of the implicitly created schema and PUBLIC is given the privilege to create objects in this schema.
- LOAD allows a user to load data into a table.

Only users with SYSADM or DBADM authority can grant and revoke these privileges to and from other users.

Note: When a database is created, the following privileges are automatically granted to PUBLIC:

- CREATETAB
- BINDADD
- CONNECT
- IMPLICIT_SCHEMA
- USE privilege on USERSPACE1 table space
- SELECT privilege on the system catalog views.

To remove any privilege, a DBADM or SYSADM must explicitly revoke the privilege from PUBLIC.

Implicit Schema Authority (IMPLICIT_SCHEMA) Considerations

When a new database is created, or when a database is migrated from the previous release, 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 (3) 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.

Schema Privileges

Schema privileges are in the object privilege category. Object privileges are shown in Figure 6 on page 250.

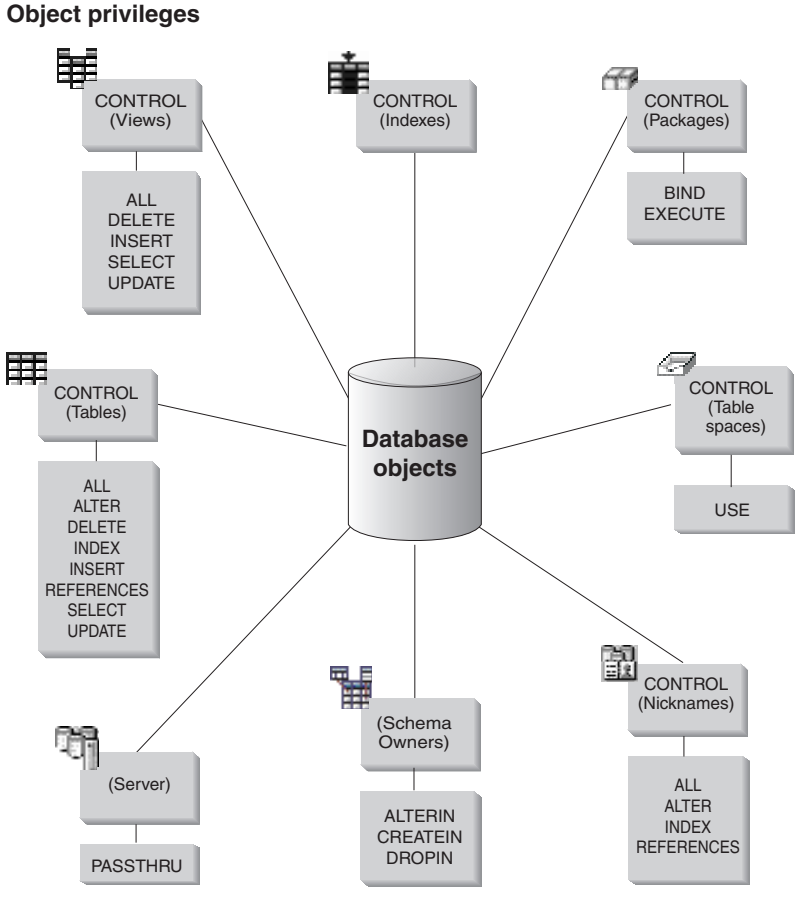


Figure 6. Object Privileges

Schema privileges involve actions on schemas in a database. A user may be granted any of the following privileges:

- CREATEIN allows the user to create objects within the schema.
- ALTERIN allows the user to alter objects within the schema.
- DROPIN allows the user to drop objects from within the schema.

The owner of the schema has all of these privileges and the ability to grant them to others. The objects that are manipulated within the schema object include: tables, views, indexes, packages, data types, functions, triggers, procedures, and aliases.

Table Space Privileges

The table space privileges involve actions on the table spaces in a database. A user may be granted the USE privilege for a table space which then allows them to create tables within the table space.

The owner of the table space, typically the creator who has SYSADM or SYSCTRL authority, has the USE privilege and the ability to grant this privilege to others. 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.

Table and View Privileges

Table and view privileges involve actions on tables or views in a database. A user must have CONNECT privilege on the database to use any of the following privileges:

- CONTROL provides the user with all privileges for a table or view including the ability to drop it, and to grant and revoke individual table privileges. You must have SYSADM or DBADM authority to grant CONTROL. The creator of a table automatically receives CONTROL privilege on the table. The creator of a view automatically receives CONTROL privilege only if they have CONTROL privilege on all tables and views referenced in the view definition, or they have SYSADM or DBADM authority.
- ALTER allows the user to add columns to a table, to add or change comments on a table and its columns, to add a primary key or unique constraint and to create or drop a table check constraint. The user can also create triggers on the table, although additional authority on all the objects referenced in the trigger (including SELECT on the table if the trigger references any of the columns of the table) is required. A user with ALTER privilege on all the descendent tables can drop a primary key; a user with ALTER privilege on the table and REFERENCES privilege on the parent table, or REFERENCES privilege on the appropriate columns, can create or drop a foreign key. A user with ALTER privilege can also COMMENT ON a table.
- DELETE allows the user to delete rows from a table or view.
- INDEX allows the user to create an index on a table. Creators of indexes automatically have CONTROL privilege on the index. For more information, see "Index Privileges" on page 255.
- INSERT allows the user to insert a row into a table or view, and to run the IMPORT utility.
- REFERENCES allows the user to create and drop a foreign key, specifying the table as the parent in a relationship. The user might have this privilege only on specific columns.
- SELECT allows the user to retrieve rows from a table or view, to create a view on a table, and to run the EXPORT utility.

- UPDATE allows the user to change an entry in a table, a view, or for one or more specific columns in a table or view. The user may have this privilege only on specific columns.

The privilege to grant these privileges to others may also be granted using the WITH GRANT OPTION on the GRANT statement.

Note: When a user or group is granted CONTROL privilege on a table, all other privileges on that table are automatically granted WITH GRANT OPTION. If you subsequently revoke the CONTROL privilege on the table from a user, that user will still retain the other privileges that were automatically granted. To revoke all the privileges that are granted with the CONTROL privilege, you must either explicitly revoke each individual privilege or specify the ALL keyword on the REVOKE statement, for example:

```
REVOKE ALL
  ON EMPLOYEE FROM USER HERON
```

When working with typed tables, there are implications regarding table and view privileges.

Note: Privileges may be granted independently at every level of a table hierarchy. As a result, a user granted a privilege on a supertable within a hierarchy of typed tables may also indirectly affect any subtables. However, a user can only operate directly on a subtable if the necessary privilege is held on that subtable.

The supertable/subtable relationships among the tables in a table hierarchy mean that operations such as SELECT, UPDATE, and DELETE will affect the rows of the operation's target table and all its subtables (if any). This behavior can be called "substitutability". For example, suppose that you have created an Employee table of type Employee_t with a subtable Manager of type Manager_t. A manager is a (specialized) kind of employee, as indicated by the type/subtype relationship between the structured types Employee_t and Manager_t and the corresponding table/subtable relationship between the tables Employee and Manager. As a result of this relationship, the SQL query:

```
SELECT * FROM Employee
```

will return the object identifier and Employee_t attributes for both employees and managers. Similarly, the update operation:

```
UPDATE Employee SET Salary = Salary + 1000
```

will give a thousand dollar raise to managers as well as regular employees.

A user with SELECT privilege on Employee will be able to perform this SELECT operation even if they do not have an explicit SELECT privilege on

Manager. However, such a user will not be permitted to perform a SELECT operation directly on the Manager subtable, and will therefore not be able to access any of the non-inherited columns of the Manager table.

Similarly, a user with UPDATE privilege on Employee will be able to perform an UPDATE operation on Manager, thereby affecting both regular employees and managers, even without having the explicit UPDATE privilege on the Manager table. However, such a user will not be permitted to perform UPDATE operations directly on the Manager subtable, and will therefore not be able to update non-inherited columns of the Manager table.

The following manuals provide information about the authorizations required to execute specific commands, APIs, or SQL statements:

- *SQL Reference*
- *Command Reference*
- *Administrative API Reference*.

Refer to *Administration Guide: Performance* for information about the authorization required to update catalog statistics.

For information about how view privileges are determined, refer to the CREATE VIEW statement in the *SQL Reference* manual.

Nickname Privileges

Nickname privileges involve actions on nicknames in a database. These privileges do not affect privileges on the data source objects referenced by nicknames. A user must have CONNECT privilege on the database to use any of the following privileges:

- CONTROL provides the user with all privileges for a nickname including the ability to drop it, and to grant and revoke individual nickname privileges. You must have SYSADM or DBADM authority to grant CONTROL. The creator of a nickname automatically receives CONTROL privilege on the nickname.
- ALTER allows the user to change column names in a nickname, add or change the DB2 type that the column's data type maps to, and set column options for nickname columns.
- INDEX allows the user to create an index specification on a nickname. Creators of index specifications automatically have CONTROL privilege on the index.
- REFERENCES allows the user to create and drop a foreign key, specifying the nickname as the parent in a relationship. The user may have this privilege only on specific columns.

The privilege to grant these privileges to others can also be granted using the WITH GRANT OPTION on the GRANT statement.

Note: When a user or group is granted CONTROL privilege on a nickname, all other privileges on that nickname are automatically granted WITH GRANT OPTION. If you subsequently revoke the CONTROL privilege on the nickname from a user, that user will still retain the other privileges that were automatically granted.

To access data source data, you must also have the proper authorization for the objects at data sources referenced by nicknames.

When a user accesses a view that references one or more nicknames, that user must be authorized to access the view and the objects that the nicknames reference at data sources.

Server Privileges

There is one server privilege: PASSTHRU. This privilege controls which authorization IDs can issue DDL and DML statements directly (pass-through operations) to data sources.

DB2 provides two SQL statements to control pass-through operations:

- GRANT PASSTHRU, which grants the authority to issue SET PASSTHRU statements against a data source and pass-through DML and DDL statements to that data source.
- REVOKE PASSTHRU, which revokes the authority to issue SET PASSTHRU statements against a data source and pass-through DML and DDL statements to that data source.

A sample statement granting pass-through authorization to the user SHAWN for the server ORACLE1 is:

```
GRANT PASSTHRU ON SERVER ORACLE1 TO USER SHAWN
```

For complete information on the syntax of PASSTHRU statements, see the *SQL Reference*.

Package Privileges

A package is a database object that contains the information needed by the database manager to access data in the most efficient way for a particular application program. Package privileges enable a user to create and manipulate packages. The user must have CONNECT privilege on the database to use any of the following privileges:

- CONTROL provides the user with the ability to rebind, drop, or execute a package as well as the ability to extend those privileges to others. The creator of a package automatically receives this privilege. A user with

CONTROL privilege is granted the BIND and EXECUTE privileges, and can grant BIND and EXECUTE privileges to other users as well. To grant CONTROL privilege, the user must have SYSADM or DBADM authority.

- BIND allows the user to rebind an existing package.
- EXECUTE allows the user to execute a package.

In addition to these package privileges, the BINDADD database privilege allows users to create new packages or rebind an existing package in the database.

Users with the authority to execute a package containing nicknames don't need additional privileges or an authority level for the nicknames within the package; however, they will need to pass authentication checks at the data sources containing the objects referenced by the nicknames. In addition, package users must have the appropriate privileges or authority levels for data source objects at the data source.

It is possible that packages containing nicknames might require additional authorization steps because DB2 uses dynamic SQL when communicating with DB2 Family data sources. The authorization ID running the package at the data source must have the appropriate authority to execute the package dynamically at that data source. See the *SQL Reference* for more information about how DB2 processes static and dynamic SQL.

Index Privileges

The creator of an index or an index specification automatically receives CONTROL privilege on the index. CONTROL privilege on an index is really the ability to drop the index. To grant CONTROL privilege on an index, a user must have SYSADM or DBADM authority.

The table-level INDEX privilege allows a user to create an index on that table (see "Table and View Privileges" on page 251).

Sequence Privileges

The creator of a sequence automatically receives the USAGE privilege. The USAGE privilege is needed to use NEXTVAL and PREVVAL expressions for the sequence. To allow other users to use the NEXTVAL and PREVVAL expressions, sequence privileges must be granted to PUBLIC. This allows all users to use the expressions with the specified sequence.

Controlling Access to Database Objects

Controlling data access requires an understanding of direct and indirect privileges, administrative authorities, and packages. This section explains these topics and provides some examples.

Directly granted privileges are stored in the system catalog. Methods for auditing the implementation of the database access control plan are discussed in "Using the System Catalog" on page 267.

Authorization is controlled in three ways:

- Explicit authorization is controlled through privileges controlled with the GRANT and REVOKE statements
- Implicit authorization is controlled by creating and dropping objects
- Indirect privileges are associated with packages.

The following topics are discussed:

- "Granting Privileges"
- "Revoking Privileges" on page 257
- "Managing Implicit Authorizations by Creating and Dropping Objects" on page 259
- "Allowing Indirect Privileges Through a Package" on page 259
- "Controlling Access to Data with Views" on page 261
- "Monitoring Access to Data Using the Audit Facility" on page 264.

Granting Privileges

The GRANT statement allows an authorized user to grant privileges. A privilege can be granted to one or more authorization names in one statement; or to PUBLIC, which makes the privileges available to all users. Note that an authorization name can be either an individual user or a group.

On operating systems where users and groups exist with the same name, you should specify whether you are granting the privilege to the user or group. Both the GRANT and REVOKE statements support the keywords USER and GROUP. If these optional keywords are not used, the database manager checks the operating system security facility to determine whether the authorization name identifies a user or a group. If the authorization name could be both a user and a group, an error is returned.

The following example grants SELECT privileges on the EMPLOYEE table to the user HERON:

```
GRANT SELECT
   ON EMPLOYEE TO USER HERON
```

The following example grants SELECT privileges on the EMPLOYEE table to the group HERON:

```
GRANT SELECT
   ON EMPLOYEE TO GROUP HERON
```

To grant privileges on most database objects, the user must have SYSADM authority, DBADM authority, or CONTROL privilege on that object; or, the user must hold the privilege WITH GRANT OPTION. Privileges can be granted only on existing objects. To grant CONTROL privilege to someone else, the user must have SYSADM or DBADM authority. To grant DBADM authority, the user must have SYSADM authority.

Refer to the *SQL Reference* for more information about the GRANT statement.

Revoking Privileges

The REVOKE statement allows authorized users to revoke privileges previously granted to other users. To revoke privileges on database objects, you must have DBADM authority, SYSADM authority, or CONTROL privilege on that object. Note that holding a privilege WITH GRANT OPTION is not sufficient to revoke that privilege. To revoke CONTROL privilege from another user, you must have SYSADM or DBADM authority. To revoke DBADM authority, you must have SYSADM authority. Privileges can only be revoked on existing objects.

Note: A user without DBADM authority or CONTROL privilege on a table or view is not able to revoke a privilege that they granted through their use of the WITH GRANT OPTION. Also, there is no cascade on the revoke to those who have received privileges granted by the person being revoked. For more information on the authority required to revoke privileges, refer to the *SQL Reference* manual.

If a privilege has been granted to both a user and a group with the same name, you must specify the GROUP or USER keyword when revoking the privilege. The following example revokes the SELECT privilege on the EMPLOYEE table from the user HERON:

```
REVOKE SELECT
   ON EMPLOYEE FROM USER HERON
```

The following example revokes the SELECT privilege on the EMPLOYEE table from the group HERON:

```
REVOKE SELECT
   ON EMPLOYEE FROM GROUP HERON
```

Note that revoking a privilege from a group may not revoke it from all members of that group. If an individual name has been directly granted a privilege, it will keep it until that privilege is directly revoked.

If a table privilege is revoked from a user, privileges are also revoked on any view created by that user which depends on the revoked table privilege.

However, only the privileges implicitly granted by the system are revoked. If a privilege on the view was granted directly by another user, the privilege is still held.

You may have a situation where you want to GRANT a privilege to a group and then REVOKE the privilege from just one member of the group. There are only a couple of ways to do that without receiving the error message SQL0556N:

- You can remove the member from the group; or, create a new group with fewer members and GRANT the privilege to the new group.
- You can REVOKE the privilege from the group and then GRANT it to individual users (authorization IDs).

If an explicitly granted table (or view) privilege is revoked from a user with DBADM authority, privileges **will not** be revoked from other views defined on that table. This is because the view privileges are available through the DBADM authority and are not dependent on explicit privileges on the underlying tables.

If you have defined a view based on one or more underlying tables or views and you lose the SELECT privilege to one or more of those tables or views, then the view cannot be used.

Note: When CONTROL privilege is revoked from a user on a table or a view, the user continues to have the ability to grant privileges to others. When given CONTROL privilege, the user also receives all other privileges WITH GRANT OPTION. Once CONTROL is revoked, all of the other privileges remain WITH GRANT OPTION until they are explicitly revoked.

All packages that are dependent on revoked privileges are marked invalid, but can be validated if rebound by a user with appropriate authority. Packages can also be rebuilt if the privileges are subsequently granted again to the binder of the application; running the application will trigger a successful implicit rebind. If privileges are revoked from PUBLIC, all packages bound by users having only been able to bind based on PUBLIC privileges are invalidated. If DBADM authority is revoked from a user, all packages bound by that user are invalidated including those associated with database utilities. Attempting to use a package that has been marked invalid causes the system to attempt to rebind the package. If this rebind attempt fails, an error occurs (SQLCODE -727). In this case, the packages must be explicitly rebound by a user with:

- Authority to rebind the packages
- Appropriate authority for the objects used within the packages

These packages should be rebound at the time the privileges are revoked. Refer to the *SQL Reference* for more information about the REVOKE and REBIND PACKAGE statements.

If you have defined a trigger based on one or more privileges and you lose one or more of those privileges, then the trigger cannot be used.

Managing Implicit Authorizations by Creating and Dropping Objects

The database manager implicitly grants certain privileges to a user who issues a CREATE SCHEMA, CREATE TABLESPACE, CREATE TABLE, CREATE VIEW, or CREATE INDEX statement, or who creates a new package using a PREP or BIND command. Privileges are also granted when objects are created by users with SYSADM or DBADM authority. Similarly, privileges are removed when an object is dropped.

When the created object is a table space, table, index, or package, the user receives CONTROL privilege on the object. When the object is a view, the CONTROL privilege for the view is granted implicitly only if the user has CONTROL privilege for all tables and views referenced in the view definition.

When the object explicitly created is a schema, the schema owner is given ALTERIN, CREATEIN, and DROPIN privileges WITH GRANT OPTION. An implicitly created schema has CREATEIN granted to PUBLIC.

For information about how view privileges are determined, refer to the CREATE VIEW statement in the *SQL Reference* manual.

Establishing Ownership of a Plan or a Package

The BIND and PRECOMPILE commands create or change an application package. On either one, use the `OWNER` option to name the owner of the resulting package. There are simple rules for naming the owner of a package:

- Any user can name themselves as the owner. This is the default if the `OWNER` option is not specified.
- An ID with SYSADM or DBADM authority can name any authorization ID as the owner using the `OWNER` option.

Not all operating systems that can bind a package using DB2 database products support the `OWNER` option.

Refer to the *Command Reference* for more information on the BIND and PRECOMPILE commands.

Allowing Indirect Privileges Through a Package

Access to data within a database can be requested by application programs, as well as by persons engaged in an interactive workstation session. A package contains statements that allow users to perform a variety of actions on many database objects. Each of these actions requires one or more privileges.

Privileges granted to individuals binding the package and to PUBLIC are used for authorization checking when static SQL is bound. Privileges granted through groups are **not** used for authorization checking when static SQL is bound. The user with a valid *authID* who binds a package must either have been explicitly granted all the privileges required to execute the static SQL statements in the package or have been implicitly granted the necessary privileges through PUBLIC unless VALIDATE RUN was specified when binding the package. If VALIDATE RUN was specified at BIND time, all authorization failures for any static SQL statements within this package will not cause the BIND to fail, and those SQL statements are revalidated at run time. PUBLIC, group, and user privileges **are all** used when checking to ensure the user has the appropriate authorization (BIND or BINDADD privilege) to bind the package.

Packages may include both static and dynamic SQL. To process a package with static SQL, a user need only have EXECUTE privilege on the package. This user can then indirectly obtain the privileges of the package binder for any static SQL in the package but only within the restrictions imposed by the package.

To process a package with any dynamic SQL statements, the user must have EXECUTE privilege on the package. The user needs EXECUTE privilege on the package plus any privileges required to execute the dynamic SQL statements in the package. The binder's authorities and privileges are used for any static SQL in the package.

Allowing Indirect Privileges Through a Package Containing Nicknames

When a package contains references to nicknames, authorization processing for package creators and package users is slightly more complex. When a package creator successfully binds packages that contain nicknames, the package creator does not have to pass authentication checking or privilege checking for the tables and views that the nicknames reference at the data source. However, the package executor must pass authentication and authorization checking at data sources.

For example, assume that a package creator's .SQC file contains several SQL statements. One static statement references a local table. Another dynamic statement references a nickname. When the package is bound, the package creator's authid is used to verify privileges for the local table, but no checking is done for the data source objects that the nickname identifies. When another user executes the package, assuming they have the EXECUTE privilege for that package, that user does not have to pass any additional privilege checking for the statement referencing the table. However, for the statement referencing the nickname, the user executing the package must pass authentication checking and privilege checking at the data source.

When the .SQC file contains only dynamic SQL statements and a mixture of table and nickname references, DB2 authorization checking for local objects and nicknames is similar. Package users must pass privilege checking for any local objects (tables, views) within the statements and also pass privilege checking for nickname objects (package users must pass authentication and privilege checking at the data source containing the objects that the nicknames identify). In both cases, users of the package must have the EXECUTE privilege.

The ID and password of the package executor is used for all data source authentication and privilege processing. This information can be changed by creating a user mapping.

Note: Nicknames cannot be specified in static SQL. Do not use the DYNAMICRULES option (set to BIND) with packages containing nicknames.

It is possible that packages containing nicknames might require additional authorization steps because DB2 uses dynamic SQL when communicating with DB2 Family data sources. The authorization ID running the package at the data source must have the appropriate authority to execute the package dynamically at that data source. See the *SQL Reference* for more information about how DB2 processes static and dynamic SQL.

Controlling Access to Data with Views

A view provides a means of controlling access or extending privileges to a table by allowing:

- Access only to designated columns of the table.

  For users and application programs that require access only to specific columns of a table, an authorized user can create a view to limit the columns addressed only to those required.
- Access only to a subset of the rows of the table.

  By specifying a WHERE clause in the subquery of a view definition, an authorized user can limit the rows addressed through a view.
- Access only to a subset of the rows or columns in data source tables or views. If you are accessing data sources through nicknames, you can create local DB2 views that reference nicknames. These views can reference nicknames from one or many data sources.

  Note: Because you can create a view that contains nickname references for more than one data source, your users can access data in multiple data sources from one view. These views are called *multi-location views*. Such views are useful when joining information in columns of sensitive tables across a distributed environment or when individual users lack the privileges needed at data sources for specific objects.

To create a view, a user must have SYSADM authority, DBADM authority, or CONTROL or SELECT privilege for each table or view referenced in the view definition. The user must also be able to create an object in the schema specified for the view. That is, CREATEIN privilege for an existing schema or IMPLICIT_SCHEMA authority on the database if the schema does not already exist. See "Creating a View" on page 144 for more information.

If you are creating views that reference nicknames, you do not need additional authority on the data source objects (tables and views) referenced by nicknames in the view; however, your users must have SELECT authority or the equivalent authorization level for the underlying data source objects when they access the view.

If your users do not have the proper authority at the data source for underlying objects (tables and views), you can:

1. Create a data source view over those columns in the data source table that are OK for the user to access
2. Grant the SELECT privilege on this view to users
3. Create a nickname to reference the view

Users can then access the columns by issuing a SELECT statement that references the new nickname.

The following scenario provides a more detailed example of how views can be used to restrict access to information.

Many people might require access to information in the STAFF table, for different reasons. For example:

- The personnel department needs to be able to update and look at the entire table.

  This requirement can be easily met by granting SELECT and UPDATE privileges on the STAFF table to the group PERSONNL:

  ```
  GRANT SELECT,UPDATE ON TABLE STAFF TO GROUP PERSONNL
  ```

- Individual department managers need to look at the salary information for their employees.

  This requirement can be met by creating a view for each department manager. For example, the following view can be created for the manager of department number 51:

  ```
  CREATE VIEW EMP051 AS
     SELECT NAME,SALARY,JOB FROM STAFF
     WHERE DEPT=51
  GRANT SELECT ON TABLE EMP051 TO JANE
  ```

  The manager with the authorization name JANE would query the EMP051 view just like the STAFF table. When accessing the EMP051 view of the

STAFF table, this manager views the following information:

NAME	SALARY	JOB
Fraye	45150.0	Mgr
Williams	37156.5	Sales
Smith	35654.5	Sales
Lundquist	26369.8	Clerk
Wheeler	22460.0	Clerk

- All users need to be able to locate other employees. This requirement can be met by creating a view on the NAME column of the STAFF table and the LOCATION column of the ORG table, and by joining the two tables on their respective DEPT and DEPTNUMB columns:

```
CREATE VIEW EMPLOCS AS
   SELECT NAME, LOCATION FROM STAFF, ORG
   WHERE STAFF.DEPT=ORG.DEPTNUMB
GRANT SELECT ON TABLE EMPLOCS TO PUBLIC
```

  Users who access the employee location view will see the following information:

NAME	LOCATION
Molinare	New York
Lu	New York
Daniels	New York
Jones	New York
Hanes	Boston
Rothman	Boston
Ngan	Boston
Kermisch	Boston
Sanders	Washington
Pernal	Washington
James	Washington
Sneider	Washington
Marenghi	Atlanta
O'Brien	Atlanta
Quigley	Atlanta
Naughton	Atlanta
Abrahams	Atlanta

NAME	LOCATION
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver
Davis	Denver
Edwards	Denver
Gafney	Denver

Monitoring Access to Data Using the Audit Facility

The DB2 audit facility generates, and allows you to maintain, an audit trail for a series of predefined database events. While not a facility that prevents access to data, the audit facility can monitor and keep a record of attempts to access or modify data objects.

SYSADM authority is required to use the audit facility administrator tool, `db2audit`.

See "Chapter 6. Auditing DB2 Activities" on page 273 for a detailed description of the DB2 audit facility.

Data Encryption

One part of your security plan may involve encrypting your data. To do this, you can use encryption and decryption built-in functions: ENCRYPT, DECRYPT_BIN, DECRYPT_CHAR, and GETHINT. For more information on these functions, including their syntax, refer to the *SQL Reference*.

The ENCRYPT function encrypts data using a password-based encryption method. These functions also allow you to encapsulate a password hint. The password hint is embedded in the encrypted data. Once encrypted, the only way to decrypt the data is by using the correct password. Developers that choose to use these functions should plan for the management of forgotten passwords and unusable data.

The result of the ENCRYPT functions is the same data type as the first argument.

Only VARCHARs can be encrypted.

The declared length of the result is one of the following:

- The length of the data argument plus 42 when the optional hint parameter is specified.
- The length of the data argument plus 10 when the optional hint parameter is not specified.

The DECRYPT_BIN and DECRYPT_CHAR functions decrypt data using password-based decryption.

The result of the DECRYPT_BIN and DECRYPT_CHAR functions is the same data type as the first argument.

The declared length of the result is the length of the original data.

The GETHINT function returns an encapsulated password hint. A password hint is a phrase that will help data owners remember passwords. For example, the word “Ocean” can be used as a hint to remember the password ″Pacific″.

The password that is used to encrypt the data is determined in one of two ways:

- Password Argument. The password is a string that is explicitly passed when the ENCRYPT function is invoked. The data is encrypted and decrypted with the given password.
- Special Register Password. The SET ENCRYPTION PASSWORD statement encrypts the password value and sends the encrypted password to the database manager to store in a special register. ENCRYPT, DECRYPT_BIN and DECRYPT_CHAR functions invoked without a password parameter use the value in the ENCRYPTION PASSWORD special register.

  The initial or default value for the special register is an empty string.

Valid lengths for passwords are between 6 and 127 inclusive. Valid lengths for hints are between 0 and 32 inclusive.

When the ENCRYPTION PASSWORD special register is set from the client, the password is encrypted at the client, sent to the database server, and then decrypted. To ensure that the password is not left readable, it is also re-encrypted at the database server. DECRYPT_BIN and DECRYPT_CHAR functions must decrypt the special register before use. The value found in the ENCRYPTION PASSWORD is also not left readable. Gateway security is not supported.

Tasks and Required Authorizations

Not all organizations divide job responsibilities in the same manner. Table 5 lists some other common job titles, the tasks that usually accompany them, and the authorities or privileges that are needed to carry out those tasks.

Table 5. Common Job Titles, Tasks, and Required Authorization

JOB TITLE	TASKS	REQUIRED AUTHORIZATION
Department Administrator	Oversees the departmental system; creates databases	SYSCTRL authority. SYSADM authority if the department has its own instance.
Security Administrator	Authorizes other users for some or all authorizations and privileges	SYSADM or DBADM authority.
Database Administrator	Designs, develops, operates, safeguards, and maintains one or more databases	DBADM and SYSMAINT authority over one or more databases. SYSCTRL authority in some cases.
System Operator	Monitors the database and carries out backup functions	SYSMAINT authority.
Application Programmer	Develops and tests the database manager application programs; may also create tables of test data	BINDADD, BIND on an existing package, CONNECT and CREATETAB on one or more databases, some specific schema privileges, and a list of privileges on some tables.
User Analyst	Defines the data requirements for an application program by examining the system catalog views	SELECT on the catalog views; CONNECT on one or more databases.
Program End User	Executes an application program	EXECUTE on the package; CONNECT on one or more databases. See the note following this table.

Table 5. Common Job Titles, Tasks, and Required Authorization (continued)

JOB TITLE	TASKS	REQUIRED AUTHORIZATION
Information Center Consultant	Defines the data requirements for a query user; provides the data by creating tables and views and by granting access to database objects	DBADM authority over one or more databases.
Query User	Issues SQL statements to retrieve, add, delete, or change data; may save results as tables	CONNECT on one or more databases; CREATEIN on the schema of the tables and views being created; and, SELECT, INSERT, UPDATE, DELETE on some tables and views.

Note: If an application program contains dynamic SQL statements, the Program End User may need other privileges in addition to EXECUTE and CONNECT (such as SELECT, INSERT, DELETE, and UPDATE).

Using the System Catalog

Information about each database is automatically maintained in a set of views called the system catalog, which is created when the database is generated. This system catalog describes tables, columns, indexes, programs, privileges, and other objects.

Six of these views list the privileges held by users and the identity of the user granting each privilege:

SYSCAT.DBAUTH Lists the database privileges

SYSCAT.TABAUTH Lists the table and view privileges

SYSCAT.COLAUTH Lists the column privileges

SYSCAT.PACKAGEAUTH Lists the package privileges

SYSCAT.INDEXAUTH Lists the index privileges

SYSCAT.SCHEMAAUTH Lists the schema privileges

SYSCAT.PASSTHRUAUTH Lists the server privilege

Privileges granted to users by the system will have SYSIBM as the grantor. SYSADM, SYSMAINT and SYSCTRL are not listed in the system catalog.

The CREATE and GRANT statements place privileges in the system catalog. Users with SYSADM and DBADM authorities can grant and revoke SELECT

privilege on the system catalog views. The following examples show how to extract information about privileges by using these SQL queries:

- "Retrieving Authorization Names with Granted Privileges"
- "Retrieving All Names with DBADM Authority"
- "Retrieving Names Authorized to Access a Table"
- "Retrieving All Privileges Granted to Users" on page 269
- "Securing the System Catalog Views" on page 270.

Retrieving Authorization Names with Granted Privileges

No single system catalog view contains information about all privileges. The following statement retrieves all authorization names with privileges:

```
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'DATABASE' FROM SYSCAT.DBAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'TABLE   ' FROM SYSCAT.TABAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'PACKAGE ' FROM SYSCAT.PACKAGEAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'INDEX   ' FROM SYSCAT.INDEXAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'COLUMN  ' FROM SYSCAT.COLAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SCHEMA  ' FROM SYSCAT.SCHEMAAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SERVER  ' FROM SYSCAT.PASSTHRUAUTH
ORDER BY GRANTEE, GRANTEETYPE, 3
```

Periodically, the list retrieved by this statement should be compared with lists of user and group names defined in the system security facility. You can then identify those authorization names that are no longer valid.

Note: If you are supporting remote database clients, it is possible that the authorization name is defined at the remote client only and not on your database server machine.

Retrieving All Names with DBADM Authority

The following statement retrieves all authorization names that have been directly granted DBADM authority:

```
SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

Retrieving Names Authorized to Access a Table

The following statement retrieves all authorization names that are directly authorized to access the table EMPLOYEE with the qualifier JAMES:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
```

```
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
```

To find out who can update the table EMPLOYEE with the qualifier JAMES, issue the following statement:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
      (CONTROLAUTH  = 'Y' OR
       UPDATEAUTH   = 'Y' OR  UPDATEAUTH   = 'G')
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
   PRIVTYPE = 'U'
```

This retrieves any authorization names with DBADM authority, as well as those names to which CONTROL or UPDATE privileges have been directly granted. However, it will not return the authorization names of users who only hold SYSADM authority.

Remember that some of the authorization names may be groups, not just individual users.

Retrieving All Privileges Granted to Users

By making queries on the system catalog views, users can retrieve a list of the privileges they hold and a list of the privileges they have granted to other users. For example, the following statement retrieves a list of the database privileges that have been directly granted to an individual authorization name:

```
SELECT * FROM SYSCAT.DBAUTH
   WHERE GRANTEE = USER AND GRANTEETYPE = 'U'
```

The following statement retrieves a list of the table privileges that were directly granted by a specific user:

```
SELECT * FROM SYSCAT.TABAUTH
   WHERE GRANTOR  = USER
```

The following statement retrieves a list of the individual column privileges that were directly granted by a specific user:

```
SELECT * FROM SYSCAT.COLAUTH
   WHERE GRANTOR  = USER
```

The keyword USER in these statements is always equal to the value of a user's authorization name. USER is a read-only special register. Refer to the *SQL Reference* for more information on special registers.

Securing the System Catalog Views

During database creation, SELECT privilege on the system catalog views is granted to PUBLIC. (See "Database Privileges" on page 248 for other privileges that are automatically granted to PUBLIC.) In most cases, this does not present any security problems. For very sensitive data, however, 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. 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 this.

At a minimum, you should consider restricting access to the following catalog views:

- SYSCAT.DBAUTH
- SYSCAT.TABAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.INDEXAUTH
- SYSCAT.COLAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.SCHEMAAUTH

This would prevent information on user privileges from becoming available to everyone with access to the database. With this information, an unethical user could gain unauthorized access to the database.

You should also examine the columns for which statistics are gathered (refer to "Catalog Statistics" in the *Administration Guide: Performance*). 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 to let each authorization name retrieve information about its own privileges.

For example, the following view MYSELECTS includes the owner and name of every table on which a user's authorization name has been directly granted SELECT privilege:

```
CREATE VIEW MYSELECTS AS
   SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABAUTH
   WHERE GRANTEETYPE = 'U'
     AND GRANTEE = USER
     AND SELECTAUTH = 'Y'
```

The keyword USER in this statement is always equal to the value of the authorization name.

The following statement makes the view available to every authorization name:

```
GRANT SELECT ON TABLE MYSELECTS TO PUBLIC
```

And finally, remember to revoke SELECT privilege on the base table:

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC
```

Chapter 6. Auditing DB2 Activities

Authentication, authorities, and privileges can be used to control known or anticipated access to data, but these methods may be insufficient to prevent unknown or unanticipated access to data. To assist in the detection of this latter type of data access, DB2 provides an audit facility. Successful monitoring of unwanted data access and subsequent analysis can lead to improvements in the control of data access and the ultimate prevention of malicious or careless unauthorized access to the data. The monitoring of application and individual user access, including system administration actions, can provide a historical record of activity on your database systems.

The DB2 audit facility generates, and allows you to maintain, an audit trail for a series of predefined database events. The records generated from this facility are kept in an audit log file. The analysis of these records can reveal usage patterns which would identify system misuse. Once identified, actions can be taken to reduce or eliminate such system misuse.

The audit facility acts at an instance level, recording all instance level activities and database level activities.

When working in a partitioned database environment, many of the auditable events occur at the partition at which the user is connected (the coordinator node) or at the catalog node (if they are not the same partition). The implication of this is that audit records can be generated by more than one partition. Part of each audit record contains information on the coordinator node and originating node identifiers.

The audit log (db2audit.log) and the audit configuration file (db2audit.cfg) are located in the instance's `security` subdirectory. At the time you create an instance, read/write permissions are set on these files, where possible, by the operating system. By default, the permissions are read/write for the instance owner only. It is recommended that you do not change these permissions.

Users of the audit facility administrator tool, db2audit, must have SYSADM authority/privileges.

The audit facility must be stopped and started explicitly. When starting, the audit facility uses existing audit configuration information. Since the audit facility is independent of the DB2 server, it will remain active even if the instance is stopped. In fact, when the instance is stopped, an audit record may be generated in the audit log.

Authorized users of the audit facility can control the following actions within the audit facility:

- Start recording auditable events within the DB2 instance.
- Stop recording auditable events within the DB2 instance.
- Configure the behavior of the audit facility, including selecting the categories of the auditable events to be recorded.
- Request a description of the current audit configuration.
- Flush any pending audit records from the instance and write them to the audit log.
- Extract audit records by formatting and copying them from the audit log to a flat file or ASCII delimited files. Extraction is done for one of two reasons: in preparation for analysis of log records or in preparation for pruning of log records.
- Prune audit records from the current audit log.

There are different categories of audit records that may be generated. In the description of the categories of events available for auditing (below), you should notice that following the name of each category is a one-word keyword used to identify the category type. The categories of events available for auditing are:

- Audit (AUDIT). Generates records when audit settings are changed or when the audit log is accessed.
- Authorization Checking (CHECKING). Generates records during authorization checking of attempts to access or manipulate DB2 objects or functions.
- Object Maintenance (OBJMAINT). Generates records when creating or dropping data objects.
- Security Maintenance (SECMAINT). Generates records when granting or revoking: object or database privileges, or DBADM authority. Records are also generated when the database manager security configuration parameters SYSADM_GROUP, SYSCTRL_GROUP, or SYSMAINT_GROUP are modified.
- System Administration (SYSADMIN). Generates records when operations requiring SYSADM, SYSMAINT, or SYSCTRL authority are performed.
- User Validation (VALIDATE). Generates records when authenticating users or retrieving system security information.
- Operation Context (CONTEXT). Generates records to show the operation context when a database operation is performed. This category allows for better interpretation of the audit log file. When used with the log's event correlator field, a group of events can be associated back to a single database operation. For example, an SQL statement for dynamic SQL, a

package identifier for static SQL, or an indicator of the type of operation being performed, such as CONNECT, can provide needed context when analyzing audit results.

Note: The SQL statement providing the operation context might be very long and is completely shown within the CONTEXT record. This can make the CONTEXT record very large.

- You can audit failures, successes, or both.

Any operation on the database may generate several records. The actual number of records generated and moved to the audit log depends on the number of categories of events to be recorded as specified by the audit facility configuration. It also depends on whether successes, failures, or both, are audited. For this reason, it is important to be selective of the events to audit.

Audit Facility Behavior

The audit facility records auditable events including those affecting database instances. For this reason, the audit facility is an independent part of DB2 that can operate even if the DB2 instance is stopped. If the audit facility is active, then when a stopped instance is started, auditing of database events in the instance resumes.

The timing of the writing of audit records to the audit log can have a significant impact on the performance of databases in the instance. The writing of the audit records can take place synchronously or asynchronously with the occurrence of the events causing the generation of those records. The value of the *audit_buf_sz* database manager configuration parameter determines when the writing of audit records is done.

If the value of this parameter is zero (0), the writing is done synchronously. The event generating the audit record will wait until the record is written to disk. The wait associated with each record causes the performance of DB2 to decrease.

If the value of *audit_buf_sz* is greater than zero, the record writing is done asynchronously. The value of the *audit_buf_sz* when it is greater than zero is the number of 4 KB pages used to create an internal buffer. The internal buffer is used to keep a number of audit records before writing a group of them out to disk. The statement generating the audit record as a result of an audit event will not wait until the record is written to disk, and can continue its operation.

In the asynchronous case, it could be possible for audit records to remain in an unfilled buffer for some time. To prevent this from happening for an

extended period, the database manager will force the writing of the audit records regularly. An authorized user of the audit facility may also flush the audit buffer with an explicit request.

There are differences when an error occurs dependent on whether there is synchronous or asynchronous record writing. In asynchronous mode there may be some records lost because the audit records are buffered before being written to disk. In synchronous mode there may be one record lost because the error could only prevent at most one audit record from being written.

The setting of the ERRORTYPE audit facility parameter controls how errors are managed between DB2 and the audit facility. When the audit facility is active, and the setting of the ERRORTYPE audit facility parameter is `AUDIT`, then the audit facility is treated in the same way as any other part of DB2. An audit record must be written (to disk in synchronous mode; or to the audit buffer in asynchronous mode) for an audit event associated with a statement to be considered successful. Whenever an error is encountered when running in this mode, a negative SQLCODE is returned to the application for the statement generating an audit record. If the error type is set to `NORMAL`, then any error from db2audit is ignored and the operation's SQLCODE is returned. See "Audit Facility Usage Scenarios" on page 277 for additional details on the ERRORTYPE audit facility parameters (and other related parameters).

Depending on the API or SQL statement and the audit settings for the DB2 instance, none, one, or several audit records may be generated for a particular event. For example, an SQL UPDATE statement with a SELECT subquery may result in one audit record containing the results of the authorization check for UPDATE privilege on a table and another record containing the results of the authorization check for SELECT privilege on a table.

For dynamic data manipulation language (DML) statements, audit records are generated for all authorization checking at the time that the statement is prepared. Reuse of those statements by the same user will not be audited again since no authorization checking takes place at that time. However, if a change has been made to one of the catalog tables containing privilege information, then in the next unit of work, the statement privileges for the cached dynamic SQL statements are checked again and one or more new audit records created.

For a package containing only static DML statements, the only auditable event that could generate an audit record is the authorization check to see if a user has the privilege to execute that package. The authorization checking and possible audit record creation required for the static SQL statements in the package is carried out at the time the package is precompiled or bound. The execution of the static SQL statements within the package is not auditable.

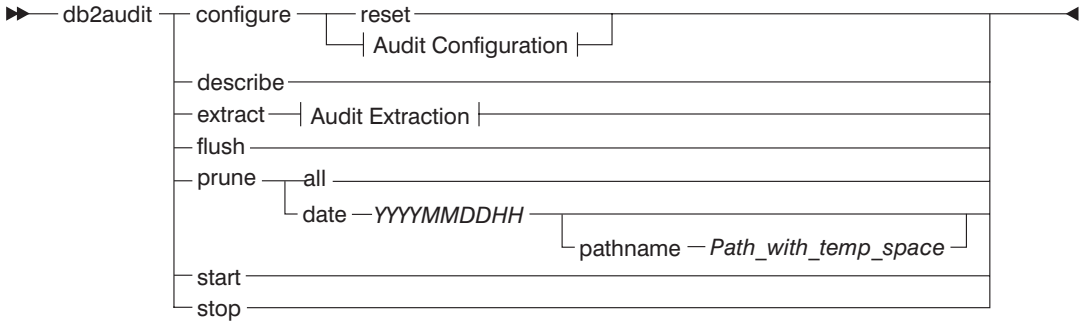
When a package is bound again either explicitly by the user, or implicitly by the system, audit records are generated for the authorization checks required by the static SQL statements.

For statements where authorization checking is performed at statement execution time (for example, data definition language (DDL), GRANT, and REVOKE statements), audit records are generated whenever these statements are used.

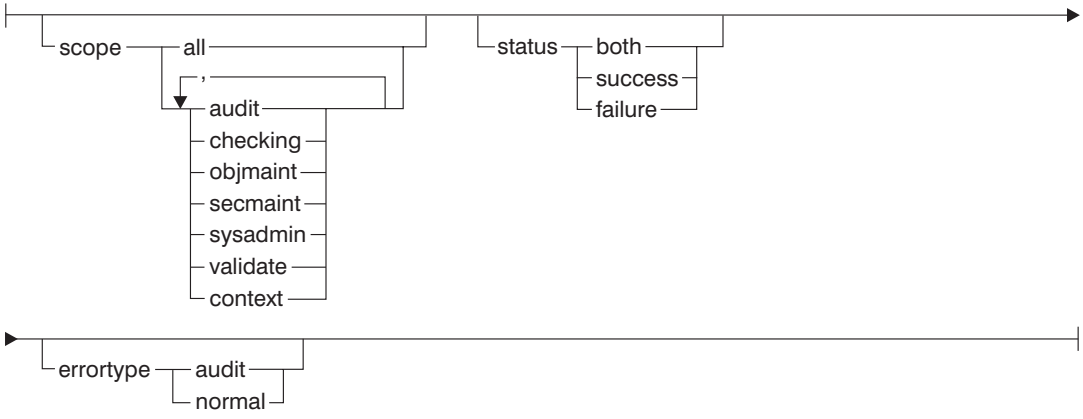
Note: When executing DDL, the section number recorded for all events (except the context events) in the audit record will be zero (0) no matter what the actual section number of the statement might have been.

Audit Facility Usage Scenarios

A review of each part of the following syntax diagrams will assist you in the understanding of how the audit facility can be used.



Audit Configuration:



Audit Extraction:

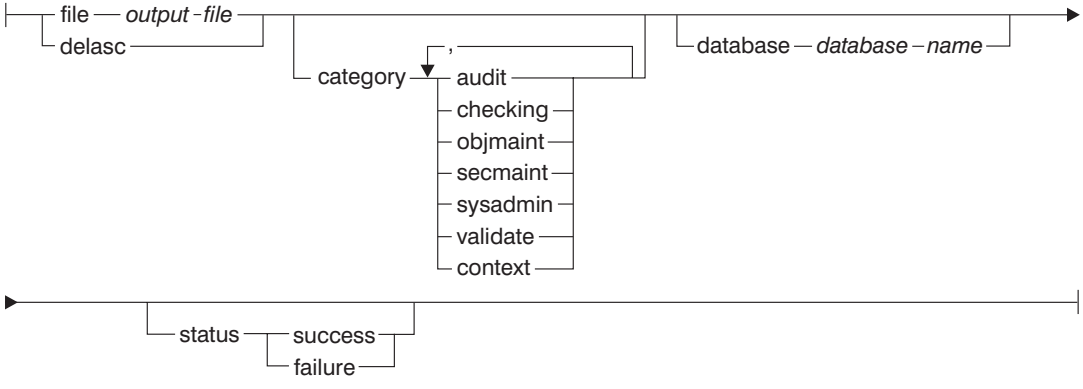


Figure 7. DB2AUDIT Syntax

The following is a description and the implied use of each parameter:

configure

This parameter allows the modification of the db2audit.cfg

configuration file in the instance's `security` subdirectory. Updates to this file can occur even when the instance is shut down. Updates occurring when the instance is active dynamically affect the auditing being done by DB2 across all partitions. The configure action on the configuration file causes the creation of an audit record if the audit facility has been started and the *audit* category of auditable events is being audited.

The following are the possible actions on the configuration file:

- RESET. This action causes the configuration file to revert to the initial configuration (where SCOPE is all of the categories except CONTEXT, STATUS is FAILURE, ERRORTYPE is NORMAL, and the audit facility is OFF). This action will create a new audit configuration file if the original has been lost or damaged.
- SCOPE. This action specifies which category or categories of events are to be audited. This action also allows a particular focus for auditing and reduces the growth of the log. It is recommended that the number and type of events being logged be limited as much as possible, otherwise the audit log will grow rapidly.

  Note: Please notice that the default SCOPE is all categories except CONTEXT and may result in records being generated rapidly. In conjunction with the mode (synchronous or asynchronous), the selection of the categories may result in a significant performance reduction and significantly increased disk requirements.
- STATUS. This action specifies whether only successful or failing events, or both successful and failing events, should be logged.

  Note: Context events occur before the status of an operation is known. Therefore, such events are logged regardless of the value associated with this parameter.
- ERRORTYPE. This action specifies whether audit errors are returned to the user or are ignored. The value for this parameter can be:
  - AUDIT. All errors including errors occurring within the audit facility are managed by DB2 and all negative SQLCODEs are reported back to the caller.
  - NORMAL. Any errors generated by db2audit are ignored and only the SQLCODEs for the errors associated with the operation being performed are returned to the application.

describe

This parameter displays to standard output the current audit configuration information and status.

extract This parameter allows the movement of audit records from the audit log to an indicated destination. If no optional clauses are specified, then all of the audit records are extracted and placed in a flat report file. If the "extract" parameter is not specified, the audit record is placed a file called *db2audit.out* in the `security` directory. If *output_file* already exists, an error message is returned.

The following are the possible options that can be used when extracting:

- FILE. The extracted audit records are placed in a file (output_file).
- DELASC. The extracted audit records are placed in a delimited ASCII format suitable for loading into DB2 relational tables. The output is placed in separate files: one for each category. The filenames are:
  - audit.del
  - checking.del
  - objmaint.del
  - secmaint.del
  - sysadmin.del
  - validate.del
  - context.del

  The DELASC choice also allows you to override the default audit character string delimiter ("0xff") when extracting from the audit log. You would use DELASC DELIMITER followed by the new delimiter that you wish to use in preparation for loading into a table that will hold the audit records. The new load delimiter can be either a single character (such as !) or a four-byte string representing a hexadecimal number (such as 0xff). For more information, refer to "Audit Facility Tips and Techniques" on page 297.
- CATEGORY. The audit records for the specified categories of audit events are to be extracted. If not specified, all categories are eligible for extraction.
- DATABASE. The audit records for a specified database are to be extracted. If not specified, all databases are eligible for extraction.
- STATUS. The audit records for the specified status are to be extracted. If not specified, all records are eligible for extraction.

flush This parameter forces any pending audit records to be written to the audit log. Also, the audit state is reset in the engine from "unable to log" to a state of "ready to log" if the audit facility is in an error state.

prune This parameter allows for the deletion of audit records from the audit

log. If the audit facility is active and the "audit" category of events has been specified for auditing, then an audit record will be logged after the audit log is pruned.

The following are the possible options that can be used when pruning:

- ALL. All of the audit records in the audit log are to be deleted.
- DATE yyyymmddhh. The user can specify that all audit records that occurred on or before the date/time specified are to be deleted from the audit log. The user may optionally supply a

  `pathname`

  which the audit facility will use as a temporary space when pruning the audit log. This temporary space allows for the pruning of the audit log when the disk it resides on is full and does not have enough space to allow for a pruning operation.

start This parameter causes the audit facility to begin auditing events based on the contents of the db2audit.cfg file. In a partitioned DB2 instance, auditing will begin on all partitions when this clause is specified. If the "audit" category of events has been specified for auditing, then an audit record will be logged when the audit facility is started.

stop This parameter causes the audit facility to stop auditing events. In a partitioned DB2 instance, auditing will be stopped on all partitions when this clause is specified. If the "audit" category of events has been specified for auditing, then an audit record will be logged when the audit facility is stopped.

Audit Facility Messages

SQL1322N An error occurred when writing to the audit log file.

Explanation: The DB2 audit facility encountered an error when invoked to record an audit event to the audit log file. There is no space on the file system where the audit log resides.

User Response: The system administrator should free up space on this file system or prune the audit log to reduce its size.

When more space is available, use db2audit to flush out any data in memory, and to reset the auditor to a ready state. Ensure that appropriate extracts have occurred, or a copy of the log has been made before pruning the log, as deleted records are not recoverable.

sqlcode: -1322

sqlstate: 50830

SQL1323N An error occurred when accessing the audit configuration file.

Explanation: The audit configuration file (db2audit.cfg) could not be opened, or was invalid. Possible reasons for this error are that the db2audit.cfg file either does not exist, or has been damaged.

User Response: Take one of the following actions:

- Restore from a saved version of the file.
- Reset the audit facility configuration file by issuing

  ```
  db2audit reset
  ```

sqlcode: -1323

sqlstate: 57019

Audit Facility Record Layouts

When an audit record is extracted from the audit log using the DELASC extract option, each record will have one of the formats shown in the following tables. Each table will begin by showing the contents of a sample record. The description of each item of the record is shown one row at a time in the associated table. If the item is important, the name of the item will be highlighted (**bold**). These items contain information that are of most interest to you.

Notes:

1. Not all fields in the sample records will have values.
2. Some fields such as "Access Attempted" are stored in the delimited ASCII format as bitmaps. In this flat report file, however, these fields will appear as a set of strings representing the bitmap values.

Table 6. Audit Record Layout for AUDIT Events

`timestamp=1998-06-24-11.54.05.151232;`**`category=AUDIT;audit event=START;`**
`event correlator=0;event status=0;`
`userid=boss;authid=BOSS;`		

NAME	**FORMAT**	**DESCRIPTION**
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: AUDIT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: CONFIGURE, DB2AUD, EXTRACT, FLUSH, PRUNE, START, STOP, and UPDATE_ADMIN_CFG
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.

Table 7. Audit Record Layout for CHECKING Events

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  object name=FOO;object type=DATABASE;
  access approval reason=DATABASE;access attempted=CONNECT;
```

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: CHECKING
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: CHECKING_OBJECT and CHECKING_FUNCTION
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator.
Application ID	VARCHAR (255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR (1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR (128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR (128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Object Schema	VARCHAR (128)	Schema of the object for which the audit event was generated.
Object Name	VARCHAR (128)	Name of object for which the audit event was generated.

Table 7. Audit Record Layout for CHECKING Events (continued)

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  object name=FOO;object type=DATABASE;
  access approval reason=DATABASE;access attempted=CONNECT;
```

NAME	FORMAT	DESCRIPTION
Object Type	VARCHAR (32)	Type of object for which the audit event was generated. Possible values include: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, BUFFERPOOL, TABLESPACE, EVENT_MONITOR, TRIGGER, DATABASE, INSTANCE, FOREIGN_KEY, PRIMARY_KEY, UNIQUE_CONSTRAINT, CHECK_CONSTRAINT, WRAPPER, SERVER, NICKNAME, USER MAPPING, SERVER OPTION, TRANSFORM, TYPE MAPPING, FUNCTION MAPPING, SUMMARY TABLES, and NONE.
Access Approval Reason	CHAR(18)	Indicates the reason why access was approved for this audit event. Possible values include: those shown in the first list following this table.
Access Attempted	CHAR(18)	Indicates the type of access that was attempted. Possible values include: those shown in the second list following this table.

The following is the list of possible CHECKING access approval reasons:

0x0000000000000001 ACCESS DENIED
Access is not approved; rather, it was denied.

0x0000000000000002 SYSADM
Access is approved; the application/user has SYSADM authority.

0x0000000000000004 SYSCTRL
Access is approved; the application/user has SYSCTRL authority.

0x0000000000000008 SYSMAINT
Access is approved; the application/user has SYSMAINT authority.

0x0000000000000010 DBADM
Access is approved; the application/user has DBADM authority.

0x0000000000000020 DATABASE PRIVILEGE
Access is approved; the application/user has an explicit privilege on the database.

0x0000000000000040 OBJECT PRIVILEGE
Access is approved; the application/user has an explicit privilege on the object or function.

0x0000000000000080 DEFINER
Access is approved; the application/user is the definer of the object or function.

0x0000000000000100 OWNER
Access is approved; the application/user is the owner of the object or function.

0x0000000000000200 CONTROL
Access is approved; the application/user has CONTROL privilege on the object or function.

0x0000000000000400 BIND
Access is approved; the application/user has bind privilege on the package.

The following is the list of possible CHECKING access attempted types:

0x0000000000000002 ALTER
Attempt to alter an object.

0x0000000000000004 DELETE
Attempt to delete an object.

0x0000000000000008 INDEX
Attempt to use an index.

0x0000000000000010 INSERT
Attempt to insert into an object.

0x0000000000000020 SELECT
Attempt to query a table or view.

0x0000000000000040 UPDATE
Attempt to update data in an object.

0x0000000000000080 REFERENCE
Attempt to establish referential constraints between objects.

0x0000000000000100 CREATE
Attempt to create an object.

0x0000000000000200 DROP
Attempt to drop an object.

0x0000000000000400 CREATEIN
Attempt to create an object within another schema.

0x0000000000000800 DROPIN
Attempt to drop an object found within another schema.

0x0000000000001000 ALTERIN
Attempt to alter or modify an object found within another schema.

0x0000000000002000 EXECUTE
Attempt to execute or run an application.

0x0000000000004000 BIND
Attempt to bind or prepare an application.

0x0000000000008000 SET EVENT MONITOR
Attempt to set event monitor switches.

0x0000000000010000 SET CONSTRAINTS
Attempt to set constraints on an object.

0x0000000000020000 COMMENT ON
Attempt to create comments on an object.

0x0000000000040000 GRANT
Attempt to grant privileges on an object to another user ID.

0x0000000000080000 REVOKE
Attempt to revoke privileges on an object from a user ID.

0x0000000000100000 LOCK
Attempt to lock an object.

0x0000000000200000 RENAME
Attempt to rename an object.

0x0000000000400000 CONNECT
Attempt to connect to an object.

0x0000000000800000 Member of SYS Group
Attempt to access or use a member of the SYS group.

0x0000000001000000 Access All
Attempt to execute a statement with all required privileges on objects held (only used for DBADM/SYSADM).

0x0000000002000000 Drop All
Attempt to drop multiple objects.

0x0000000004000000 LOAD
Attempt to load a table in a table space.

0x0000000008000000 USE
Attempt to create a table in a table space.

Table 8. Audit Record Layout for OBJMAINT Events

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
  event correlator=3;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;
```

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: OBJMAINT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: CREATE_OBJECT, RENAME_OBJECT, and DROP_OBJECT
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR (255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR (1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR (128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR (128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Object Schema	VARCHAR (128)	Schema of the object for which the audit event was generated.
Object Name	VARCHAR (128)	Name of object for which the audit event was generated.

Table 8. Audit Record Layout for OBJMAINT Events (continued)

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
  event correlator=3;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;
```

NAME	FORMAT	DESCRIPTION
Object Type	VARCHAR (32)	Type of object for which the audit event was generated. Possible values include: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, BUFFERPOOL, TABLESPACE, EVENT_MONITOR, TRIGGER, DATABASE, INSTANCE, FOREIGN_KEY, PRIMARY_KEY, UNIQUE_CONSTRAINT, CHECK_CONSTRAINT, WRAPPER, SERVER, NICKNAME, USER MAPPING, SERVER OPTION, TRANSFORM, TYPE MAPPING, FUNCTION MAPPING, SUMMARY TABLES, and NONE.

Table 9. Audit Record Layout for SECMAINT Events

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: SECMAINT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: GRANT, REVOKE, IMPLICIT_GRANT, IMPLICIT_REVOKE, and UPDATE_DBM_CFG.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.

Table 9. Audit Record Layout for SECMAINT Events (continued)

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

NAME	FORMAT	DESCRIPTION
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR (255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR (1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR (128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR (128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Object Schema	VARCHAR (128)	Schema of the object for which the audit event was generated.
Object Name	VARCHAR (128)	Name of object for which the audit event was generated.
Object Type	VARCHAR (32)	Type of object for which the audit event was generated. Possible values include: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, BUFFERPOOL, TABLESPACE, EVENT_MONITOR, TRIGGER, DATABASE, INSTANCE, FOREIGN_KEY, PRIMARY_KEY, UNIQUE_CONSTRAINT, CHECK_CONSTRAINT, WRAPPER, SERVER, NICKNAME, USER MAPPING, SERVER OPTION, TRANSFORM, TYPE MAPPING, FUNCTION MAPPING, SUMMARY TABLES, and NONE.
Grantor	VARCHAR (128)	Grantor ID.
Grantee	VARCHAR (128)	Grantee ID for which a privilege or authority was granted or revoked.
Grantee Type	VARCHAR (32)	Type of the grantee that was granted to or revoked from. Possible values include: USER, GROUP, or BOTH.

Table 9. Audit Record Layout for SECMAINT Events (continued)

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

NAME	FORMAT	DESCRIPTION
Privilege or Authority	CHAR(18)	Indicates the type of privilege or authority granted or revoked. Possible values include: Those shown in the list following this table.

The following is the list of possible SECMAINT privileges or authorities:

0x0000000000000001 Control Table
Control privilege granted or revoked on a table.

0x0000000000000002 ALTER TABLE
Privilege granted or revoked to alter a table.

0x0000000000000004 ALTER TABLE with GRANT
Privilege granted or revoked to alter a table with granting of privileges allowed.

0x0000000000000008 DELETE TABLE
Privilege granted or revoked to drop a table.

0x0000000000000010 DELETE TABLE with GRANT
Privilege granted or revoked to drop a table with granting of privileges allowed.

0x0000000000000020 Table Index
Privilege granted or revoked on an index.

0x0000000000000040 Table Index with GRANT
Privilege granted or revoked on an index with granting of privileges allowed.

0x0000000000000080 Table INSERT
Privilege granted or revoked on an insert on a table.

0x0000000000000100 Table INSERT with GRANT
Privilege granted or revoked on an insert on a table with granting of privileges allowed.

0x0000000000000200 Table SELECT
Privilege granted or revoked on a select on a table.

0x0000000000000400 Table SELECT with GRANT
Privilege granted or revoked on a select on a table with granting of privileges allowed.

0x0000000000000800 Table UPDATE
Privilege granted or revoked on an update on a table.

0x0000000000001000 Table UPDATE with GRANT
Privilege granted or revoked on an update on a table with granting of privileges allowed.

0x0000000000002000 Table REFERENCE
Privilege granted or revoked on a reference on a table.

0x0000000000004000 Table REFERENCE with GRANT
Privilege granted or revoked on a reference on a table with granting of privileges allowed.

0x0000000000008000 Package BIND
BIND privilege granted or revoked on a package.

0x0000000000010000 Package EXECUTE
EXECUTE privilege granted or revoked on a package.

0x0000000000020000 CREATEIN Schema
CREATEIN privilege granted or revoked on a schema.

0x0000000000040000 CREATEIN Schema with GRANT
CREATEIN privilege granted or revoked on a schema with granting of privileges allowed.

0x0000000000080000 DROPIN Schema
DROPIN privilege granted or revoked on a schema.

0x0000000000100000 DROPIN Schema with GRANT
DROPIN privilege granted or revoked on a schema with granting of privileges allowed.

0x0000000000200000 ALTERIN Schema
ALTERIN privilege granted or revoked on a schema.

0x0000000000400000 ALTERIN Schema with GRANT
ALTERIN privilege granted or revoked on a schema with granting of privileges allowed.

0x0000000000800000 DBADM Authority
DBADM authority granted or revoked.

0x0000000001000000 CREATETAB Authority
Createtab authority granted or revoked.

0x0000000002000000 BINDADD Authority
Bindadd authority granted or revoked.

0x0000000004000000 CONNECT Authority
CONNECT authority granted or revoked.

0x0000000008000000 Create not fenced Authority
Create not fenced authority granted or revoked.

0x0000000010000000 Implicit Schema Authority
Implicit schema authority granted or revoked.

0x0000000020000000 Server PASSTHRU
Privilege granted or revoked to use the pass-through facility with this server (federated database data source).

0x0000000100000000 Table Space USE
Privilege granted or revoked to create a table in a table space.

0x0000000200000000 Table Space USE with GRANT
Privilege granted or revoked to create a table in a table space with granting of privileges allowed.

0x0000000400000000 Column UPDATE
Privilege granted or revoked on an update on one or more specific columns of a table.

0x0000000800000000 Column UPDATE with GRANT
Privilege granted or revoked on an update on one or more specific columns of a table with granting of privileges allowed.

0x0000001000000000 Column REFERENCE
Privilege granted or revoked on a reference on one or more specific columns of a table.

0x0000002000000000 Column REFERENCE with GRANT
Privilege granted or revoked on a reference on one or more specific columns of a table with granting of privileges allowed.

0x0000004000000000 LOAD Authority
LOAD authority granted or revoked.

Table 10. Audit Record Layout for SYSADMIN Events

```
timestamp=1998-06-24-11.54.04.129923;category=SYSADMIN;audit event=DB2AUDIT;
  event correlator=1;event status=0;
  userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155404;application name=db2audit;
```

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: SYSADMIN

Table 10. Audit Record Layout for SYSADMIN Events (continued)

```
timestamp=1998-06-24-11.54.04.129923;category=SYSADMIN;audit event=DB2AUDIT;
  event correlator=1;event status=0;
  userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155404;application name=db2audit;
```

NAME	FORMAT	DESCRIPTION
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: Those shown in the list following this table.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR (255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR (1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR (128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR (128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.

The following is the list of possible SYSADMIN audit events:

Table 11. SYSADMIN Audit Events

START_DB2
STOP_DB2
CREATE_DATABASE
DROP_DATABASE
UPDATE_DBM_CFG
UPDATE_DB_CFG
CREATE_TABLESPACE
DROP_TABLESPACE
ALTER_TABLESPACE
RENAME_TABLESPACE
CREATE_NODEGROUP
DROP_NODEGROUP
ALTER_NODEGROUP
CREATE_BUFFERPOOL
DROP_BUFFERPOOL
ALTER_BUFFERPOOL
CREATE_EVENT_MONITOR
DROP_EVENT_MONITOR
ENABLE_MULTIPAGE
MIGRATE_DB_DIR
DB2TRC
DB2SET
ACTIVATE_DB
ADD_NODE
BACKUP_DB
CATALOG_NODE
CATALOG_DB
CATALOG_DCS_DB
CHANGE_DB_COMMENT
DEACTIVATE_DB
DROP_NODE_VERIFY
FORCE_APPLICATION
GET_SNAPSHOT
LIST_DRDA_INDOUBT_TRANSACTIONS
MIGRATE_DB
RESET_ADMIN_CFG
RESET_DB_CFG
RESET_DBM_CFG
RESET_MONITOR
RESTORE_DB
ROLLFORWARD_DB
SET_RUNTIME_DEGREE
SET_TABLESPACE_CONTAINERS
UNCATALOG_DB
UNCATALOG_DCS_DB
UNCATALOG_NODE
UPDATE_ADMIN_CFG
UPDATE_MON_SWITCHES
LOAD_TABLE
DB2AUDIT
SET_APPL_PRIORITY
CREATE_DB_AT_NODE
KILLDBM
MIGRATE_SYSTEM_DIRECTORY
DB2REMOT
DB2AUD
MERGE_DBM_CONFIG_FILE
UPDATE_CLI_CONFIGURATION
OPEN_TABLESPACE_QUERY
SINGLE_TABLESPACE_QUERY
CLOSE_TABLESPACE_QUERY
FETCH_TABLESPACE
OPEN_CONTAINER_QUERY
FETCH_CONTAINER_QUERY
CLOSE_CONTAINER_QUERY
GET_TABLESPACE_STATISTICS
DESCRIBE_DATABASE
ESTIMATE_SNAPSHOT_SIZE
READ_ASYNC_LOG_RECORD
PRUNE_RECOVERY_HISTORY
UPDATE_RECOVERY_HISTORY
QUIESCE_TABLESPACE
UNLOAD_TABLE
UPDATE_DATABASE_VERSION
CREATE_INSTANCE
DELETE_INSTANCE
SET_EVENT_MONITOR
GRANT_DBADM
REVOKE_DBADM
GRANT_DB_AUTHORITIES
REVOKE_DB_AUTHORITIES
REDIST_NODEGROUP

Table 12. Audit Record Layout for VALIDATE Events

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;audit event=CHECK_GROUP_MEMBERSHIP;
  event correlator=2;event status=-1092;
  database=FOO;userid=boss;authid=BOSS;execution id=newton;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  auth type=SERVER;
```

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: VALIDATE
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: GET_GROUPS, GET_USERID, AUTHENTICATE_PASSWORD, and VALIDATE_USER.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	INTEGER	Status of audit event, represented by an SQLCODE where Successful event > = 0 Failed event < 0
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Execution ID	VARCHAR(1024)	Execution ID in use at the time of the audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR (255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR (1024)	Application name in use at the time the audit event occurred.
Authentication Type	VARCHAR (32)	Authentication type at the time of the audit event.
Package Schema	VARCHAR (128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR (128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.

Table 13. Audit Record Layout for CONTEXT Events

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;audit event=EXECUTE_IMMEDIATE;
  event correlator=3;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=203;text=create table audit(c1 char(10), c2 integer);
```

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: CONTEXT
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: Those shown in the list following this table.
Event Correlator	INTEGER	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Database Name	CHAR(8)	Name of the database for which the event was generated. Blank if this was an instance level audit event.
User ID	VARCHAR(1024)	User ID at time of audit event.
Authorization ID	VARCHAR(128)	Authorization ID at time of audit event.
Origin Node Number	SMALLINT	Node number at which the audit event occurred.
Coordinator Node Number	SMALLINT	Node number of the coordinator node.
Application ID	VARCHAR (255)	Application ID in use at the time the audit event occurred.
Application Name	VARCHAR (1024)	Application name in use at the time the audit event occurred.
Package Schema	VARCHAR (128)	Schema of the package in use at the time of the audit event.
Package Name	VARCHAR (128)	Name of package in use at the time the audit event occurred.
Package Section Number	SMALLINT	Section number in package being used at the time the audit event occurred.
Statement Text (statement)	CLOB (32K)	Text of the SQL statement, if applicable. Null if no SQL statement text is available.

The following is the list of possible CONTEXT audit events:

Table 14. CONTEXT Audit Events

CONNECT	SET_APPL_PRIORITY
CONNECT_RESET	RESET_DB_CFG
ATTACH	GET_DB_CFG
DETACH	GET_DFLT_CFG
DARI_START	UPDATE_DBM_CFG
DARI_STOP	SET_MONITOR
BACKUP_DB	GET_SNAPSHOT
RESTORE_DB	ESTIMATE_SNAPSHOT_SIZE
ROLLFORWARD_DB	RESET_MONITOR
OPEN_TABLESPACE_QUERY	OPEN_HISTORY_FILE
FETCH_TABLESPACE	CLOSE_HISTORY_FILE
CLOSE_TABLESPACE_QUERY	FETCH_HISTORY_FILE
OPEN_CONTAINER_QUERY	SET_RUNTIME_DEGREE
CLOSE_CONTAINER_QUERY	UPDATE_AUDIT
FETCH_CONTAINER_QUERY	DBM_CFG_OPERATION
SET_TABLESPACE_CONTAINERS	DISCOVER
GET_TABLESPACE_STATISTIC	OPEN_CURSOR
READ_ASYNC_LOG_RECORD	CLOSE_CURSOR
QUIESCE_TABLESPACE	FETCH_CURSOR
LOAD_TABLE	EXECUTE
UNLOAD_TABLE	EXECUTE_IMMEDIATE
UPDATE_RECOVERY_HISTORY	PREPARE
PRUNE_RECOVERY_HISTORY	DESCRIBE
SINGLE_TABLESPACE_QUERY	BIND
LOAD_MSG_FILE	REBIND
UNQUIESCE_TABLESPACE	RUNSTATS
ENABLE_MULTIPAGE	REORG
DESCRIBE_DATABASE	REDISTRIBUTE
DROP_DATABASE	COMMIT
CREATE_DATABASE	ROLLBACK
ADD_NODE	REQUEST_ROLLBACK
FORCE_APPLICATION	IMPLICIT_REBIND

Audit Facility Tips and Techniques

In most cases, when working with CHECKING events, the object type field in the audit record is the object being checked to see if the required privilege or authority is held by the user ID attempting to access the object. For example, if a user attempts to ALTER a table by adding a column, then the CHECKING event audit record will indicate the access attempted was “ALTER” and the object type being checked was “TABLE” (note: not the column since it is table privileges that must be checked).

However, when the checking involves verifying if a database authority exists to allow a user ID to CREATE or BIND an object, or to delete an object, then

although there is a check against the database, the object type field will specify the object being created, bound, or dropped (rather than the database itself).

When creating an index on a table, the privilege to create an index is required, therefore the CHECKING event audit record will have an access attempt type of “index” rather than “create”.

When binding a package that already exists, then an OBJMAINT event audit record is created for the DROP of the package and then another OBJMAINT event audit record is created for the CREATE of the new copy of the package.

SQL Data Definition Language (DDL) may generate OBJMAINT or SECMAINT events that are logged as successful. It is possible however that following the logging of the event, a subsequent error may cause a ROLLBACK to occur. This would leave the object as not created; or the GRANT or REVOKE actions as incomplete. The use of CONTEXT events becomes important in this case. Such CONTEXT event audit records, especially the statement that ends the event, will indicate the nature of the completion of the attempted operation.

When extracting audit records in a delimited ASCII format suitable for loading into a DB2 relational table, you should be clear regarding the delimiter used within the statement text field. This can be done when extracting the delimited ASCII file and is done using:

```
db2audit extract delasc delimiter <load delimiter>
```

The *load delimiter* can be a single character (such as ") or a four-byte string representing a hexadecimal value (such as “0xff”). Examples of valid commands are:

```
db2audit extract delasc
db2audit extract delasc delimiter !
db2audit extract delasc delimiter 0xff
```

If you have used anything other than the default load delimiter (“"”) as the delimiter when extracting, you should use the MODIFIED BY option on the LOAD command. A partial example of the LOAD command with “0xff” used as the delimiter follows:

```
db2 load from context.del of del modified by chardel0xff replace into ...
```

This will override the default load character string delimiter which is “0xff”.

Controlling DB2 Audit Facility Activities

As part of our discussion on the control of the audit facility activities, we will use a simple scenario: A user, *newton*, runs an application called *testapp* that connects and creates a table. This same application is used in each of the examples discussed below.

We begin by presenting an extreme example: You have determined to audit all successful and unsuccessful audit events, therefore you will configure the audit facility in the following way:

```
db2audit configure scope all status both
```

Note: This creates audit records for every possible auditable event. As a result, many records are written to the audit log and this reduces the performance of your database manager. This extreme case is shown here for demonstration purposes only; there is no recommendation that you configure the audit facility with the command shown above.

After beginning the audit facility with this configuration (using "db2audit start"), and then running the *testapp* application, the following records are generated and placed in the audit log. By extracting the audit records from the log, you will see the following records generated for the two actions carried out by the application:

Action Type of Record Created

CONNECT

```
timestamp=1998-06-24-08.42.10.555345;category=CONTEXT;
audit event=CONNECT;event correlator=2;database=FOO;
application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.10.944374;category=VALIDATE;
audit event=AUTHENTICATION;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;execution id=newton;
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;

timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
```

```
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
object name=FOO;object type=DATABASE;access approval reason=DATABASE;
access attempted=CONNECT;

timestamp=1998-06-24-08.42.11.801554;category=CONTEXT;
audit event=COMMIT;event correlator=2;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;
```

CREATE TABLE

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;
audit event=EXECUTE_IMMEDIATE;event correlator=3;database=FOO;
userid=boss;authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;package name=SQLC28A1;
package section=203;text=create table audit(c1 char(10), c2 integer);

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;

timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;
audit event=CREATE_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
```

```
timestamp=1998-06-24-08.42.42.018900;category=CONTEXT;audit event=COMMIT;
event correlator=3;database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;
```

As you can see, there are a significant number of audit records generated from the audit configuration that requests the auditing of all possible audit events and types.

In most cases, you will configure the audit facility for a more restricted or focused view of the events you wish to audit. For example, you may want to only audit those events that fail. In this case, the audit facility could be configured as follows:

```
db2audit configure scope audit,checking,objmaint,secmaint,sysadmin,
   validate status failure
```

Note: This configuration is the initial audit configuration or the one that occurs when the audit configuration is reset.

After beginning the audit facility with this configuration, and then running the *testapp* application, the following records are generated and placed in the audit log. (And we assume *testapp* has not been run before.) By extracting the audit records from the log, you will see the following records generated for the two actions carried out by the application:

Action Type of Record Created

CONNECT

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;
```

CREATE TABLE

```
(none)
```

The are far fewer audit records generated from the audit configuration that requests the auditing of all possible audit events (except CONTEXT) but only when the event attempt fails. By changing the audit configuration you can control the type and nature of the audit records that are generated.

The audit facility can allow you to create audit records when those you want to audit have been successfully granted privileges on an object. In this case, you could configure the audit facility as follows:

```
db2audit configure scope checking status success
```

After beginning the audit facility with this configuration, and then running the *testapp* application, the following records are generated and placed in the audit log. (And we assume *testapp* has not been run before.) By extracting the audit records from the log, you will see the following records generated for the two actions carried out by the application:

Action Type of Record Created

CONNECT

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;
```

CREATE TABLE

```
(none)
```

Part 4. Moving Data

Chapter 7. Utilities for Moving Data

The LOAD utility moves data into tables, extends existing indexes, and generates statistics. LOAD moves the data much faster than the IMPORT utility when large amounts of data are involved. Data, unloaded using the EXPORT utility, can be loaded with the LOAD utility.

The AutoLoader utility splits large amounts of data and loads the split data into the different partitions of a partitioned database.

The IMPORT and EXPORT utilities move data between a table or view and another database or spreadsheet program; between DB2 databases; and between DB2 databases and host databases using DB2 Connect.

Data Replication (formerly DataPropagator Relational (DPROPR)) is a component of DB2 Universal Database that allows automatic copying of table updates to other tables in other DB2 relational databases.

Note: All of the information on these topics, and the comparable topics from the *Command Reference* and the *Administrative API Reference*, have been consolidated into the *Data Movement Utilities Guide and Reference*.

The *Data Movement Utilities Guide and Reference* is your primary, single source of information for these topics.

To find out more about replication, see *Replication Guide and Reference*.

Part 5. Recovery

Chapter 8. Recovering a Database

You will need to be able to recover your database when things go wrong. Problems may include power failures, application failures, as well as media and storage failures. To ensure that you can recover when problems like this happen, you need to have backups or copies of the entire database or of the table spaces that make up the database. These backups can then be used following a database problem to restore the database.

The rebuilding of a database following a problem is called recovery. Crash recovery automatically attempts to recover the database after a failure. Crash recovery protects a database from being left in an inconsistent or unusable state. Transactions against the database can be left incomplete when a failure of the database occurs. Crash recovery either rolls back incomplete transactions or commits completed transactions. These actions make the database consistent and usable.

There are two other types of recovery when the database has been damaged and you are unsure about the contents of the database. The two ways to recover a damaged database are: version recovery and roll-forward recovery. If you are working with a read-only database or where you are not concerned about transactions being recorded in a database, then you may only need version recovery. If you have taken a backup of the database, then you can apply or restore that copy of the database. This is called version recovery. If you are working with a database that has transactions applied to the database and you need to know that the database has all of those changes applied, then you need to complete a roll-forward recovery. A roll-forward recovery involves restoring a backup of the database. Then, you must apply a record of the logs recording the transactions against the database. The application of the logs repeats all of the activity against the database so that the database is brought to a state just prior to the point of failure. No changes to the database are lost using this recovery method. Logging is the key to this recovery method.

Note: All of the information on these topics, and the comparable topics from the *Command Reference* and the *Administrative API Reference*, have been consolidated into the *Data Recovery and High Availability Guide and Reference*.

The *Data Recovery and High Availability Guide and Reference* is your primary, single source of information for these topics.

Part 6. Appendixes

Appendix A. Naming Rules

Go to the section that describes the naming rules that you require information on:

- "General Naming Rules"
- "Object Naming Rules"
- "How Case-Sensitive Values Are Preserved in a Federated System" on page 317

General Naming Rules

Unless otherwise specified, all names can include the following characters:

- A through Z. When used in most names, characters A through Z are converted from lowercase to uppercase.
- 0 through 9
- @, #, $, and _ (underscore)

Names cannot begin with a number or with the underscore character.

Do not use SQL reserved words to name tables, views, columns, indexes, or authorization IDs. For a list of SQL reserved words, see the *SQL Reference*.

There are other special characters that might work separately depending on your operating system and where you are working with DB2. However, while they might work, there is no guarantee that they will work. It is not recommended that you use these other special characters when naming objects in your database.

Object Naming Rules

All objects follow the General Naming Rules. In addition, some objects have additional restrictions shown below.

Table 15. Database, Database Alias and Instance Naming Rules

Objects	Guidelines
• Databases • Database aliases • Instances	• Database names must be unique within the location in which they are cataloged. On UNIX-based implementations of DB2, this location is a directory path, while on Windows implementations, it is a drive letter. • Database alias names must be unique within the system database directory. When a new database is created, the alias defaults to the database name. As a result, you cannot create a database using a name that exists as a database alias, even if there is no database with that name. • Database, database alias and instance names can have up to 8 bytes. • On Windows NT and Windows 2000 systems, no instance can have the same name as a service name. **Note:** To avoid potential problems, do not use the special characters @, #, and $ in a database name if you intend to use the database in a communications environment. Also, because these characters are not common to all keyboards, do not use them if you plan to use the database in another language.

Table 16. Database Object Naming Rules

Objects	Guidelines
• Aliases • Buffer pools • Columns • Event monitors • Indexes • Methods • Nodegroups • Schemas • Stored procedures • Tables • Table spaces • Triggers • UDFs • UDTs • Views	Can contain up to 18 bytes *except* for the following: • Table names (including view names, summary table names, alias names, and correlation names), which can contain up to 128 bytes • Column names, which can contain up to 30 bytes • Schema names, which can contain up to 30 bytes • Object names can also include: – valid accented characters (such as ö) – multibyte characters, except multibyte spaces (for multibyte environments)

Additional Information about Schema Names

- Tables with schema names longer than 18 bytes cannot be replicated.
- User-defined types (UDTs) cannot have schema names longer than 8 bytes.

- The following schema names are reserved words and must not be used: SYSCAT, SYSFUN, SYSIBM, SYSSTAT.
- To avoid potential migration problems in the future, do not use schema names that begin with SYS. The database manager will not allow you to create triggers, user-defined types or user-defined functions using a schema name beginning with SYS.
- It is recommended that you not use SESSION as a schema name. Declared temporary tables must be qualified by SESSION. It is therefore possible to have an application declare a temporary table with a name identical to that of a persistent table, in which case the application logic can become overly complicated. Avoid the use of the schema SESSION, except when dealing with declared temporary tables.

Table 17. User, User ID and Group Naming Rules

Objects	Guidelines
• Group names • User names • User IDs	• Group names can contain up to 8 bytes. • User IDs on UNIX-based systems can contain up to 8 characters. • User names on Windows can contain up to 30 characters. Windows NT and Windows 2000 currently have a practical limit of 20 characters. • When using DCE authentication, user names have a limit of 8 characters. • When not using DCE or Client authentication, non-Windows 32-bit clients connecting to Windows NT and Windows 2000 with user names longer than 8 characters are supported when the user name and password are specified explicitly. • Names and IDs cannot: – Be USERS, ADMINS, GUESTS, PUBLIC, LOCAL or any SQL reserved word listed in the *SQL Reference*. – Begin with IBM, SQL or SYS. – Include accented characters. On UNIX-based systems, groups and users can have the same name. For the GRANT statement, you must specify whether you are referring to a group or to a user. For the REVOKE statement, specifying user or group depends on whether or not there are multiple rows in the authorization catalog tables for the GRANTEE with different values of GRANTEETYPE. On Windows NT, local groups, global groups, and users cannot have the same name. On OS/2, groups and users cannot have the same name. **Notes:** 1. Some operating systems allow case sensitive user IDs and passwords. You should check your operating system documentation to see if this is the case. 2. The authorization ID returned from a successful CONNECT or ATTACH is truncated to 8 characters. An ellipsis (...) is appended to the authorization ID and the SQLWARN fields contain warnings to indicate truncation. For more information, see the CONNECT statement in the *SQL Reference*.

Additional Information about Passwords

You may be required to perform password maintenance tasks. Since such tasks are required at the server, and many users are not able or comfortable working with the server environment, performing these tasks can pose a significant challenge. DB2 UDB provides a way to update and verify passwords without having to be at the server. For example, DB2 for OS/390 Version 5 supports this method of changing a user's password. If an error

message SQL1404N "Password expired" is received, use the CONNECT statement to change the password as follows:

```
CONNECT TO <database> USER <userid> USING <password>
   NEW <new_password> CONFIRM <new_password>
```

The "Password change" dialog of the DB2 Client Configuration Assistant (CCA) can also be used to change the password. For more information about these methods of changing the password, refer to the *SQL Reference* and the CCA online help.

Table 18. Federated Database Object Naming Rules

Objects	Guidelines
• Function mappings • Index specifications • Nicknames • Servers • Type mappings • User mappings • Wrappers	• Nicknames, mappings, index specifications, servers, and wrapper names cannot exceed 128 bytes. • Server and nickname options and option settings are limited to 255 bytes. • Names for federated database objects can also include: – Valid accented letters (such as ö) – Multibyte characters, except multibyte spaces (for multibyte environments)

Using Delimited Identifiers in Object Names

Keywords can be used. If a keyword is used in a context where it could also be interpreted as an SQL keyword, it must be specified as a delimited identifier.

Using delimited identifiers, it is possible to create an object that violates these naming rules; however, subsequent use of the object could result in errors. For example, if you create a column with a + or – sign included in the name and you subsequently use that column in an index, you will experience problems when you attempt to reorganize the table. For information about delimited identifiers, see the "SQL Identifiers" section of the *SQL Reference*.

How Case-Sensitive Values Are Preserved in a Federated System

With distributed requests, you sometimes need to specify identifiers and passwords that are case sensitive at the data source. To ensure that the case is correct when they are passed to the data source, follow these guidelines:

- Specify identifiers and passwords in the required case, and enclose them in double quotation marks.
- If you are specifying a user ID, set the *fold_id* server option to "n" ("No, don't change case") for the data source. If you are specifying a password, set the *fold_pw* server option to "n" for the data source.

There is an alternative for user IDs and passwords. If a data source requires a user ID to be in lowercase, you can specify it in any case and set the *fold_id* server option to "l" ("Send this ID to the data source in lowercase"). If the data source requires the ID to be in uppercase, you can specify it in any case and set *fold_id* to "u" ("Send this ID to the data source in uppercase"). In the same way, if a data source requires a password to be in lowercase or uppercase, you can meet this requirement by setting the *fold_pw* server option to "l" or "u".

For more information about server options, see "Using Server Options to Help Define Data Sources and Facilitate Authentication Processing" on page 153.

- If you enclose a case sensitive identifier or a password in double quotation marks at an operating system command prompt, you must ensure that the system parses the double quotation marks correctly. To do this:
  - On a UNIX-based operating system, enclose the statement in single quotation marks.
  - On the Windows NT operating system, precede each quotation mark with a backslash.

  For example, many delimited identifiers in DB2 data sources are case sensitive. Suppose you want to create a nickname, `NICK1`, for a DB2 for CS view, `"my_schema"."wkly_sal"`, that resides in a data source called `NORBASE`.

  At the command prompt for a UNIX-based system, you would type:

  ```
  db2 'create nickname nick1 for norbase."my_schema"."wkly_sal"'
  ```

  At a Windows NT command prompt, you would type:

  ```
  db2 create nickname nick1 for norbase.\"my_schema\".\"wkly_sal\"
  ```

  If you enter the statement from a DB2 command prompt (interactive mode), or if you specify it in an application program, you do not need the single quotation marks or the backslashes. For example, at the DB2 command prompt on either a UNIX-based system or Windows NT, you would type:

  ```
  create nickname nick1 for norbase."my_schema"."wkly_sal"
  ```

Appendix B. Using Distributed Computing Environment (DCE) Directory Services

DCE provides the Cell Directory Service (CDS) and Global Directory Service (GDS). For more information about DCE concepts and these services, refer to the *Introduction to OSF DCE* manual. The DB2 function for DCE Directory Services supports CDS only. With this support, the user does not have to create each database, node, and DCS database on every single client. All of this information is centralized in DCE CDS.

The following sections describe how to set up and access a database using DCE Directory Services:

- Creating Directory Objects
- Attributes of Each Object Class
- Directory Services Security
- Configuration Parameters and Registry Variables
- CATALOG and ATTACH Commands, and the CONNECT Statement
- How a Client Connects to a Database
- How Directories Are Searched
- Temporarily Overriding DCE Directory Information
- Directory Services Tasks
- Directory Services Restrictions

DCE directory services may not be supported by all DB2 clients. If DCE directory services is supported for a DB2 client, your *Quick Beginnings* manual provides additional information.

Creating Directory Objects

There are three types of directory objects that the database administrator needs to create:

- "Database Objects" on page 320
- "Database Locator Objects" on page 321
- "Routing Information Objects" on page 322

Each object contains attributes. Refer to "Attributes of Each Object Class" on page 324 for a complete description of the attributes.

Before the database administrator can create the objects, the DCE administrator needs to add database information into a CDS table and grant create privileges to the database administrator. Refer to "DCE Administrator Tasks" on page 340 for the details.

Database Objects

A database object is required for each target database. The object has a name that contains the cell name concatenated to the directory name and the name of the database, for example:

```
/.../cell_name/dir_name1/dir_name2/OBJ_NAME
```

Note: The following is recommended for the name of the database. The name should be less than or equal to 8 characters and all the characters should be upper case. If the name is mixed case or longer than 8 characters, you need to use the CATALOG GLOBAL DATABASE command to assign an alias. See "CATALOG GLOBAL DATABASE Command" on page 332 for details about the command.

The following is an example of a database object. The object stored in the DCE directory contains other information such as a timestamp. The letter to the left of each attribute indicates if the attribute is required - R, optional - O, or a comment - C.

```
   Object name:              /.../CELL_TORONTO/subsys/database/AIXDB1
R  DB_Object_Type:           D
C  DB_Product_Name:          DB2_for_AIX
C  DB_Product_Release:       V7R1M000
R  DB_Native_Database_Name:  AIXDBASE
R  DB_Database_Protocol:     DB2RA
R  DB_Authentication:        CLIENT
O  DB_Communication_Protocol:
O  DB_Database_Locator_Name: /.../CELL_TORONTO/subsys/database/AIX_INST
C  DB_Comment:               Test_database_on_AIX
```

If the database is one of many databases associated with a database manager instance, the database object should contain the name of a database locator object and the communication protocol should be blank. The name of the database locator object is the fully-qualified name of the database manager or DB2 Connect instance.

Here is an example of the DCE commands to create the object. Before any objects can be created, the DCE administrator needs to do the steps described in "DCE Administrator Tasks" on page 340.

First you must type the following in a file called *cdscp.inp*:

```
create object /.:/subsys/database/AIXDB1

add object /.:/subsys/database/AIXDB1 DB_Object_Type           = D
add object /.:/subsys/database/AIXDB1 DB_Product_Name          = DB2_for_AIX
```

```
add object /.:/subsys/database/AIXDB1 DB_Product_Release        = V7R1M000
add object /.:/subsys/database/AIXDB1 DB_Native_Database_Name   = AIXDBASE
add object /.:/subsys/database/AIXDB1 DB_Database_Protocol      = DB2RA
add object /.:/subsys/database/AIXDB1 DB_Authentication         = CLIENT
add object /.:/subsys/database/AIXDB1 DB_Database_Locator_Name  = /...
/CELL_TORONTO/subsys/database/AIX_INST
add object /.:/subsys/database/AIXDB1 DB_Comment      = Test_database_on_AIX
```

Then you must run either

- dcelogin principal password (on OS/2); or,
- dce_login principal password (on UNIX, Windows operating systems).

This should be followed by

- cdscp < cdscp.inp

Use the following command to display the object:

```
cdscp show object /.:/subsys/database/AIXDB1
```

If the database is the only database associated with a database manager instance, the database object should contain values for the Communication Protocol attribute and the name of the database locator object should be blank. For example:

```
   Object name:                 /.../CELL_TORONTO/subsys/database/MVSDB
R  DB_Object type:              D
C  DB_Product_Name:             DB2_for_MVS
C  DB_Product_Release:          V7R1M00
R  DB_Native_Database_Name:     MVSDBASE
R  DB_Database_Protocol:        DRDA
R  DB_Authentication:           SERVER
O  DB_Communication_Protocol:   APPC;NET1;TARGETLU1;DB2DRDA;MODE1;PROGRAM
O  DB_Database_Locator_Name:
C  DB_Comment:                  Test_database_on_MVS
```

Database Locator Objects

These objects contain the details about all the communication protocols used by a database management system instance or a DB2 Connect instance. One database locator object is required for:

- Each instance with both a database management system and DB2 Connect
- Each database management system instance which is associated with more than one database, but without an associated DB2 Connect
- Each DB2 Connect instance without an associated database management system.

The object has a name that contains the cell name concatenated to the directory name and the one-part name of the database instance, for example:

```
/.../cell_name/dir_name1/dir_name2/AIX_INST
```

Note: If the instance is used as the target of an ATTACH, the one-part name must be less than or equal to 8 characters and all upper case.

The following is an example of a database locator object. The object stored in the DCE directory contains other information such as a timestamp. The letter to the left of each attribute indicates if the attribute is required - R, optional - O, or a comment - C.

```
   Object name:               /.../CELL_TORONTO/subsys/database/AIX_INST
R  DB_Object_Type:            L
C  DB_Product_Name:           DB2_for_AIX
C  DB_Product_Release:        V7R1M00
R  DB_Communication_Protocol: TCPIP;HOSTNAME1;1234
R  DB_Communication_Protocol: APPC;NET1;TARGETLU1;TPN1;MODE;PROGRAM
C  DB_Comment:                Test_instance_on_AIX
```

When an attribute is defined in both the database object and the database locator object, the value in the database object is used.

Here is an example of the DCE commands to create the object. Before any objects can be created, the DCE administrator needs to do the steps described in "DCE Administrator Tasks" on page 340.

First you must type the following in a file called *cdscp.inp*:

```
create object /.:/subsys/database/AIX_INST

add object /.:/subsys/database/AIX_INST DB_Object_Type            = L
add object /.:/subsys/database/AIX_INST DB_Product_Name           = DB2_for_AIX
add object /.:/subsys/database/AIX_INST DB_Product_Release        = V7R1M00
add object /.:/subsys/database/AIX_INST DB_Communication_Protocol = TCPIP;
HOSTNAME1;1234
add object /.:/subsys/database/AIX_INST DB_Communication_Protocol = APPC;NET1;
TARGETLU;TPN1;MODE;PROGRAM
add object /.:/subsys/database/AIX_INST DB_Comment     = Test_instance_on_AIX
```

Then you must run either

- dcelogin principal password (on OS/2); or,
- dce_login principal password (on UNIX, Windows operating systems).

This should be followed by

- cdscp < cdscp.inp

Use the following command to display the object:

```
cdscp show object /.:/subsys/database/AIX_INST
```

Routing Information Objects

Routing information objects are required for host access. When a mismatch exists in the database protocol used by a client and the database protocol used by the target database, the routing object tells the client which DB2 Connect

instance to use. Attributes exist for each target database, which include the database protocols that are available and the name of the database locator object for the DB2 Connect instance. The object has a name that contains the cell name concatenated to the directory name and a unique one-part name, for example:

```
/.../cell_name/dir_name1/dir_name2/ROUTE1
```

The following is an example of a routing information object. The object stored in the DCE directory contains other information such as a timestamp. The letter to the left of each attribute indicates if the attribute, and each token within an attribute is required - R, optional - O, or a comment - C.

Client group 1 is Client_1, Client_2, and Client_3 in Figure 8 on page 334.

```
   Object name:    /.../CELL_TORONTO/subsys/database/ROUTE1
R  DB_Object_Type: R
C  DB_Comment:     Routing_for_client_group_1

R  DB_Target_Database_Info
   R  Database name                 = /.../CELL_TORONTO/subsys/database/MVSDB
   R  Outbound protocol from router = DRDA
   R  Inbound protocol to router    = DB2RA
   R  Authenticate at gateway       = 1
   O  Parameter string              = NOMAP,D,INTERRUPT_ENABLED
   R  DB_Database_Locator_Name      = /.../CELL_TORONTO/subsys/database/GW_INST

R  DB_Target_Database_Info
   R  Database name                 = *OTHERDBS
   R  Outbound protocol from router = DRDA
   R  Inbound protocol to router    = DB2RA
   R  Authenticate at gateway       = 0
   O  Parameter string              =
   R  DB_Database_Locator_Name      = /.../CELL_TORONTO/subsys/database/OTH_INST
```

The database name *OTHERDBS is a special value that identifies a common router used to access any target database not explicitly defined in the routing information object.

Here is an example of the DCE commands to create the object. The backslash (\) character is a continuation character.

Before any objects can be created, the DCE administrator needs to do the steps described in “DCE Administrator Tasks” on page 340.

First you must type the following in a file called *cdscp.inp*:

```
create object /.:/subsys/database/ROUTE1

add object /.:/subsys/database/ROUTE1 DB_Object_Type = R
add object /.:/subsys/database/ROUTE1 DB_Comment = Routing_for_client_group_1
add object /.:/subsys/database/ROUTE1 DB_Target_Database_Info = \
```

```
/.../CELL_TORONTO/subsys/database/MVSDB;\
drda;db2ra;1;NOMAP,D,INTERRUPT_ENABLE;\
/.../CELL_TORONTO/subsys/database/GW_INST
add object /.:/subsys/database/ROUTE1 DB_Target_Database_Info = \
*OTHERDBS;drda;db2ra;0;;\
/.../CELL_TORONTO/subsys/database/OTH_INST
```

Then you must run either

- dcelogin principal password (on OS/2); or,
- dce_login principal password (on UNIX, Windows operating systems).

This should be followed by

- cdscp < cdscp.inp

Use the following command to display the object:

```
cdscp show object /.:/subsys/database/ROUTE1
```

For more information about the DCE commands, refer to the following OSF DCE publications:

- *OSF DCE Administration Guide*
- *OSF DCE Administration Reference*

Attributes of Each Object Class

In the DCE environment, each object and object attribute is identified by an object ID (OID). Each OID is obtained from a hierarchy of allocation authorities, where the highest authority is the International Organization for Standardization (ISO).

Table 19 shows the attributes for each object class and Table 20 on page 325 shows their attributes.

Table 19. Object Attribute Classes

Object Class	Object ID (OID)	Required Attributes	Optional Attributes
(DB) Database_Object	1.3.18.0.2.6.12	DAU, DOT, DDP, DNN	DCO, DPN, DRL, DLN, DCP, DPR
(DL) Database_Locator_Object	1.3.18.0.2.6.13	DOT, DCP	DCO, DPN, DRL
(RI) Routing_Information_Object	1.3.18.0.2.6.14	DOT, DTI	DCO, DPN, DRL

Table 20. Object Class Attributes

Attribute Name	OID	Minimum Length	Maximum Length	Syntax
(DAU) DB_Authentication	1.3.18.0.2.4.39	1	1024	Char
(DCO) DB_Comment	1.3.18.0.2.4.30	1	1024	Char
(DCP) DB_Communication_Protocol	1.3.18.0.2.4.31	1	1024	Char
(DDP) DB_Database_Protocol	1.3.18.0.2.4.32	1	1024	Char
(DLN) DB_Database_Locator_Name	1.3.18.0.2.4.33	1	1024	Char
(DNN) DB_Native_Database_Name	1.3.18.0.2.4.34	1	1024	Char
(DOT) DB_Object_Type	1.3.18.0.2.4.35	1	1	Char
(DPN) DB_Product_Name	1.3.18.0.2.4.36	1	1024	Char
(DRL) DB_Product_Release	1.3.18.0.2.4.37	1	1024	Char
(DTI) DB_Target_Database_Info	1.3.18.0.2.4.38	1	1024	Char
(DPR) DB_Principal	1.3.18.0.2.4.63	1	1024	Char
Note: Multiple values are allowed for DCP, DDP, and DTI. Only one value is allowed for the other attributes.				

Details About Each Attribute

The following section describes each attribute.

Note: DCE Directory Services does not check that the entries are valid for DB2. Ensure that you enter the attributes that are required and that you enter the correct values.

DB_Authentication (DAU)
Authentication method required by the object. This attribute is required for the database object of a DB2 server. The value must be CLIENT, SERVER, or DCE.

DB_Principal (DPR)
If authentication method is "DCE", enter the DCE principal in this attribute.

DB_Comment (DCO)
For documentation purposes only.

DB_Communication_Protocol (DCP)
A multi-value attribute where each value consists of tokens that describe the network protocol supported. Examples of the network protocols are TCP/IP, APPC, IPX/SPX, and NetBIOS. Each token is separated by a semicolon. Do not put spaces between the tokens.

- The tokens for TCP/IP are:
  1. tcpip

2. Host name of the target node
3. Port number used by the object to listen for incoming TCP/IP connect requests
4. (Optional) Security can be either NONE or SOCKS.

For example: `tcpip;HOSTNAME;1234`

- The tokens for APPC are:
  1. appc
  2. Network ID of the target to which to object belongs.
  3. LU name where the target can be found.
  4. Transaction Program Name (TPN) representing the object in the LU (For DB2 for MVS/ESA, use DB2DRDA as the TPN.)
  5. Mode name
  6. Type of security used by the target. The values are:
     - NONE
     - PROGRAM
     - SAME

  For example: `appc;NETID;TARGETLU;TPNAME;MODE;PROGRAM`

  Note: For APPC, the client must use its local control point (CP) as its LU name.
- (OS/2 and supported Windows operating systems only) The tokens for IPX/SPX are:
  1. ipxspx
  2. Name of the file server
  3. Name of the object

  For example: `ipxspx;SVR_NAME;OBJ_NAME`
- (OS/2 and supported Windows operating systems only) The tokens for NetBIOS are:
  1. netbios
  2. Node name of the server

  For example: `netbios;SVR_NNME` where the client adapter number is found in either the registry value *db2clientadpt* or the database manager configuration parameter *dft_client_adpt*.
- (Supported Windows operating systems only) The tokens for NPIPE are:
  1. NPIPE

2. Computer name of the server
3. Instance name of the server

For example: `npipe;computername;instance`

DB_Database_Protocol (DDP)
The database protocol or protocols supported by the target database. Examples of the values are DB2RA and DRDA. The following are the *cdscp* commands to add two protocols.

```
add object /.:/subsys/database/AIXDB1 DB_Database_Protocol db2ra
add object /.:/subsys/database/AIXDB1 DB_Database_Protocol drda
```

DB_Database_Locator_Name (DLN)
The DCE name of the database locator object. In the database object, the name is for the DBMS instance. In the routing information object, the name is for the DB2 Connect instance.

For example, `/.../CELL_TORONTO/subsys/database/AIX_INST`

DB_Native_Database_Name (DNN)
The database name or alias by which the database is known within the instance containing the database. This is the name that a local application on the instance uses to connect to that database.

The name is up to 8 characters for a DB2 for Universal Database database. For other databases, the length of the name may be different. For example it can be up to 18 characters for databases on DB2 for MVS/ESA.

DB_Object_Type (DOT)
The type of object. This attribute is required for all objects and can be one of the following:

D Database object

L Database locator object

R Routing information object

DB_Product_Name (DPN)
The identification of the product. For documentation purposes only.

DB_Product_Release (DRL)
The product release level. For documentation purposes only.

DB_Target_Database_Info (DTI)
A multi-value attribute where each value consists of a fixed number of tokens, separated by a semicolon. Do not put spaces between the tokens. The tokens must be in the following order:

1. Database name. The DCE name of a target database for which the routing service is provided. The value *OTHERDBS specifies a default gateway for any target databases not explicitly defined in the routing information object.
2. Outbound protocol from router. The database protocol used by the target database, or the database protocol the routing DB2 Connect instance uses to communicate with that target database. For example, DRDA.
3. Inbound protocol to router. The database protocol accepted by the routing DB2 Connect instance object. For example, DB2RA.
4. Authenticate at gateway. The valid values are 0 or 1. See Table 21 on page 329 for more details.
5. Parameter string which contains information specific to the DB2 Connect gateway. The string contains tokens that must be in the order described below. The tokens are separated by commas. For tokens that are not specified, the default is used.
   - Map-file name. The fully-qualified name of the SQLCODE mapping file that overrides the default SQLCODE mapping. To turn off SQLCODE mapping, specify NOMAP.
   - D. The application disconnects from the DRDA server database when specific SQLCODEs are returned. Refer to the *DB2 Connect User's Guide* for details about the SQLCODEs.
   - INTERRUPT_ENABLED. DB2 Connect will drop the connection and roll back the unit of work when a client issues an interrupt while connected to the DRDA server.

   The following are some examples:

   ```
   NOMAP
   /u/username/sqllib/map/dcs1new.map,D
   /u/username/sqllib/map/dcs1new.map,D,INTERRUPT_ENABLED
   ```

   Where defaults are used, use a comma to preserve the order of the tokens, for example:

   ```
   ,D
   ```

   or

   ```
   ,,INTERRUPT_ENABLED
   ```

   Refer to the *DB2 Connect User's Guide* for details about the Parameter string.
6. The DCE name of the DB2 Connect instance that provides the routing service.

The following is an example of the DB_Target_Database_Info:

```
/.../CELL_TORONTO/subsys/database/MVSDB;\
drda;db2ra;0;;\
/.../CELL_TORONTO/subsys/database/GW_INST
```

Note: In the above example, the back slash (\) is a line continuation character.

Directory Services Security

When using DCE directory services in an environment without a DB2 Connect gateway, authentication is the same as is used for other clients accessing database servers. For more information, see "Selecting an Authentication Method for Your Server" on page 225.

When using DCE directory services in an environment with a DB2 Connect gateway, the DB2 Connect administrator determines where user names and passwords are validated. With DCE directories, specify the following:

- The security type of the communication protocol in the database locator object representing the DB2 Connect workstation. (If a remote client is connected to the DB2 Connect Extended Edition gateway via an APPC connection, specify a security type of NONE in the *DCE Locator Object* of the gateway.)
- The authentication type in the database object.
- The security type of the communication protocol in the database object (or its associated locator object).
- The authenticate at gateway token in the routing information object.

Table 21 shows the possible combinations of these values and where validation is performed for each combination using APPC connections. The combinations shown in this table are supported by DB2 Connect with DCE Directory Services.

Table 21. Valid Security Scenarios with DCE using APPC Connections

	Database Object of the Server		**Routing Object**	**Validation**
Case	**Authentication**	**Security**	**Authenticate at Gateway**	
1	CLIENT	SAME	0	Remote client (or DB2 Connect workstation)
2	CLIENT	SAME	1	DB2 Connect workstation
3	SERVER	PROGRAM	0	DRDA server
4	SERVER	PROGRAM	1	DB2 Connect workstation and DRDA server
5	DCE	NONE	NOT APPLICABLE	DCE

Table 22 shows the possible combinations of these values and where validation is performed for each combination using TCP/IP connections. The combinations shown in this table are supported by DB2 Connect with DCE Directory Services.

Table 22. Valid Security Scenarios with DCE using TCP/IP Connections

Case	Authentication	Authenticate at Gateway	Validation
1	CLIENT	0	Client
2	CLIENT	1	DB2 Connect workstation
3	SERVER	0	DRDA server
4	NOT APPLICABLE	NOT APPLICABLE	None
5	DCE	NOT APPLICABLE	DCE

Each combination is applicable to both APPC and TCP/IP and is described in more detail below:

1. The user name and password are validated only at the remote client. (For a local client, the user name and password are validated only at the DB2 Connect workstation.)

   The user is expected to be authenticated at the location he or she first signs on to. The user ID is sent across the network, but not the password. Use this type of security only if all client workstations have adequate security facilities.
2. The user name and password are validated at the DB2 Connect workstation only. The password is sent across the network from the remote client to the DB2 Connect workstation but not to the DRDA server.
3. The user name and password are validated at the DRDA server only. The password is sent across the network from the remote client to the DB2 Connect workstation and from the DB2 Connect workstation to the DRDA server.
4. The user name and password are validated at both the DB2 Connect workstation and the DRDA server. The password is sent across the network from the remote client to the DB2 Connect workstation and from the DB2 Connect workstation to the DRDA server.

   Because validation is performed in two places, the same set of user names and passwords must be maintained at both the DB2 Connect workstation and the DRDA server.
5. A DCE token is obtained from the DCE Security Server.

Notes:

1. For AIX-based systems, all users using security type `SAME` must belong to the AIX system group.

2. For AIX-based systems with remote clients, the instance of the DB2 Connect product running on the DB2 Connect workstation must belong to the AIX system group.
3. Access to a DRDA server is controlled by its own security mechanisms or subsystems; for example, the Virtual Telecommunications Access Method (VTAM) and Resource Access Control Facility (RACF). Access to protected database objects is controlled by the SQL **GRANT** and **REVOKE** statements.

Configuration Parameters and Registry Variables

The following configuration parameters are used with DCE directories. An example of the values is shown. Refer to "Distributed Services" within the chapter "Configuring DB2" in *Administration Guide: Performance* for details.

- *dir_obj_name* is the database instance name which is concatenated with *dir_path_name*. If the instance name is used as the target of the ATTACH command, the name must be less than or equal to 8 characters and all upper case, for example:

  ```
  AIX_INST
  ```

- *dir_type* identifies whether or not to use DCE directory services. To enable DCE directory services, this parameter must be set to:

  ```
  DCE
  ```

  Note that *dir_type* is set to NONE and cannot be updated on database clients that do not support the use of DCE directory services.
- *dir_path_name* is the directory path name provided by the DCE administrator, for example:

  ```
  /.:/subsys/database/
  ```

- *route_obj_name* is an optional parameter that provides the DCE directory services name of the routing information object. The name can be fully-qualified, for example:

  ```
  /.:/subsys/database/ROUTE1
  ```

  or a one-part name that will be concatenated with *dir_path_name*, for example:

  ```
  ROUTE1
  ```

- *dft_client_comm* is an optional DCE parameter that specifies the communications protocol used by the client, for example:

  ```
  TCPIP
  ```

  This parameter can also specify more than one protocol, for example:

  ```
  TCPIP,APPC   (on UNIX-based platforms)
  TCPIP,APPC,IPXSPX,NETBIOS    (on OS/2 platforms)
  TCPIP,APPC,IPXSPX,NETBIOS,NPIPE    (on supported Windows operating systems)
  ```

- *dft_client_adpt* is an optional DCE parameter that specifies the default client adapter number for the NetBIOS protocol on OS/2 and supported Windows operating systems. The valid range of numbers is zero through fifteen (0 to 15). If this parameter contains a non-numeric value, then the value defaults to zero (0). If this parameter contains a value outside the range allowed, then the value defaults to zero (0).

For the following parameters, registry variables can override the parameter values.

Configuration Parameter	Registry Variable
dir_path_name	DB2DIRPATHNAME
route_obj_name	DB2ROUTE
dft_client_comm	DB2CLIENTCOMM
dft_client_adpt	DB2CLIENTADPT

The rules for setting these registry variables is the same as their corresponding configuration parameter. For example, like the *dft_client_comm* parameter, the DB2CLIENTCOMM is a character string that can have multiple values, each separated by a comma, for example:

```
db2set DB2CLIENTCOMM=TCPIP,APPC
```

CATALOG and ATTACH Commands, and the CONNECT Statement

DCE information needs to be specified in the following commands:

- CATALOG GLOBAL DATABASE Command
- CONNECT Statement
- ATTACH Command

CATALOG GLOBAL DATABASE Command

Use the CATALOG GLOBAL DATABASE command when the client and server have a different path name, or when the database name contains more than 8 characters or mixed case characters. The database administrator enters the DCE name of the database and directory type DCE.

For example:

- When the path names are different, for example if *dir_path_name* = */.../CELL_TORONTO/subsys/database/*:

```
CATALOG GLOBAL DATABASE
/.../CELL_VANCOUVER/subsys/database/VMDB AS VANVMDB
USING DIRECTORY DCE WITH "comment-string"
```

- When the database name contains more than 8 characters, such as the name DB_LONGNAME:

```
CATALOG GLOBAL DATABASE
/.../CELL_VANCOUVER/subsys/database/DB_LONGNAME AS VANVMDB
USING DIRECTORY DCE WITH "comment-string"
```

CONNECT Statement

To retrieve the appropriate DCE directory object, the client must know the fully-qualified DCE name of the database or the DBMS instance. Some of the methods of specifying the name in the CONNECT statement follow.

- Enter the alias, for example:

  ```
  CONNECT TO VANVMDB
  ```

- Enter the one-part name, for example:

  ```
  CONNECT TO VMDB
  ```

In this case, the path name specified at the client must be the same as the path name specified at the server. (The path name is specified by the *dir_path_name* configuration parameter or the corresponding registry value.)

ATTACH Command

The effective path name of the client must be the same as the path name of the target DBMS instance.

If the *dir_path_name* is the same for client and server (for example, `/.../CELL_TORONTO/subsys/database/`) and the *dir_obj_name* at the database server is `AIX_INST`, the command to attach to the instance is:

```
ATTACH TO AIX_INST
```

How a Client Connects to a Database

Figure 8 on page 334 shows a sample configuration of a database network with two DCE cells. `/.../CELL_TORONTO` and `/.../CELL_VANCOUVER` are the names of the cells. (Each of these cells contains a directory called `/.:/subsys/database/` and while not illustrated in diagram, is used in other examples.)

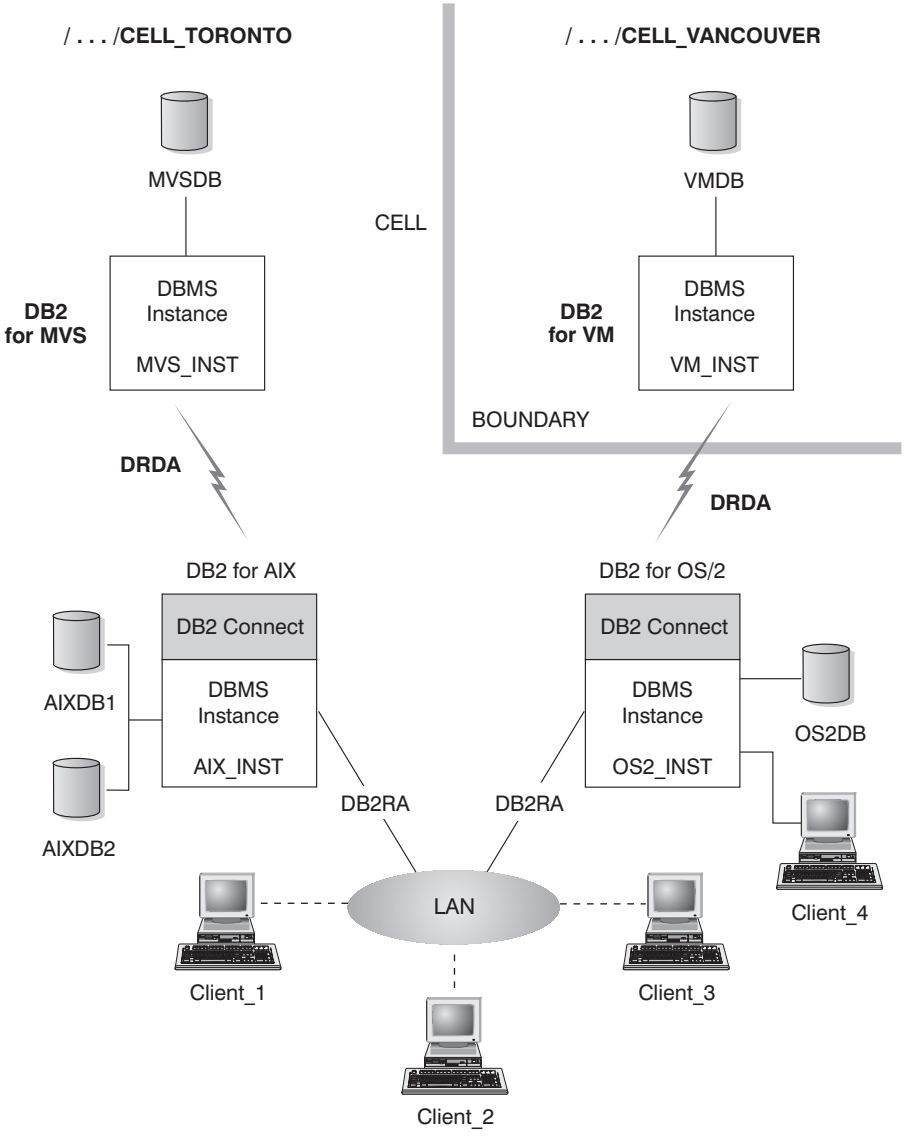


Figure 8. Configuration of A Network Database

To allow the clients in the TORONTO cell to access all the databases in both cells, values must be specified in the database manager configuration parameters and the following objects must be created:

- A database object for each database.
- A database locator object for the target databases on the two database servers for DB2 for AIX and DB2 for OS/2.

- A single routing information object that is known to all clients. The attributes specify which DB2 Connect node to use for the MVSDB and VMDB databases.

The following provide examples of how a client connects to a database:

- Connecting to Databases in the Same Cell
- Connecting to a Database in a Different Cell.

These examples include the database manager configuration parameters that must be specified.

Connecting to Databases in the Same Cell

This section describes several examples of how clients connect to databases in the same cell.

1. Client_1 connects to AIXDB2. The database shares the same directory path name as the client.

   The database administrator needs to:

   - Specify the directory path name value in the configuration parameter *dir_path_name* (or the DB2DIRPATHNAME registry value).
   - Specify the directory services type value to be DCE in the configuration parameter *dir_type*.
   - Specify the communication protocol in the configuration parameter *dft_client_comm* (or the DB2CLIENTCOMM registry value).

   The local system database directory does not contain AIXDB2, so the DCE directory is searched using the fully-qualified name. The name is created by concatenating the value for the configuration parameter *dir_path_name* (or the DB2DIRPATHNAME registry value) with AIXDB2.

   The sequence of events is:

   a. Client_1 obtains the database object for AIXDB2 using the DCE name of the database `/.../CELL_TORONTO/subsys/database/AIXDB2`.
   b. From this object, Client_1 knows that AIXDB2 uses the DB protocol DB2RA, which is the same protocol that Client_1 uses.
   c. The DB protocols match, so Client_1 reads the database management system locator object for AIX_INST, retrieves the communications protocol attribute value that matches the one it uses, and uses the information to start a conversation with that database management system instance.

2. Client_3 connects to MVSDB. The database shares the same directory path name as the client and uses a different database protocol from the client.

   The database administrator needs to:

- Specify the directory path name value in the configuration parameter *dir_path_name* (or the DB2DIRPATHNAME registry value).
- Specify the directory services type value to be DCE in the configuration parameter *dir_type*.
- Specify the communication protocol in the configuration parameter *dft_client_comm* (or the DB2CLIENTCOMM registry value).
- Specify the DCE name of the default routing information object in the configuration parameter *route_obj_name* (or the DB2ROUTE registry value).

The sequence of events is:

a. Client_3 obtains the database object for MVSDB using the DCE name of the database `/.../CELL_TORONTO/subsys/database/MVSDB`.
b. From this object, Client_3 finds that MVSDB only uses the DB protocol DRDA, which is not the protocol that Client_3 uses.
c. Client_3 then obtains the routing information object using the name defined in the *route_obj_name* configuration parameter or the DB2ROUTE registry value. The client finds the target database information for MVSDB.
d. Client_3 reads the database locator object associated with the MVSDB target database information, retrieves the communication protocol, and sends an SQL CONNECT request to the router.
e. The router then sets up an APPC connection with MVSDB.

Connecting to a Database in a Different Cell

This section describes an example of how a client connects to a database in a different cell when the database protocols are different.

1. Client_3 has previously been configured to use the following:
   - DCE directory services, by specifying DCE for the *dir_type* parameter.
   - A cell other than CELL_VANCOUVER through the configuration parameter *dir_path_name*, for example:

     ```
     /.../CELL_TORONTO/subsys/database/
     ```
2. In order for Client_3 to connect to VMDB, the database administrator needs to:
   - Explicitly catalog VMDB in the local system database directory. Associate the DCE name for VMDB with a locally unique database alias, and issue the CONNECT statement with the alias value. For example:

     ```
     CATALOG GLOBAL DATABASE
     /.../CELL_VANCOUVER/subsys/database/VMDB AS VANVMDB
     USING DIRECTORY DCE WITH "comment-string"
     ```

     followed by:

     ```
     CONNECT TO VANVMDB
     ```

- Specify the communication protocol in the configuration parameter *dft_client_comm* (or the DB2CLIENTCOMM registry value).
- Specify the DCE name of the default routing information object in the configuration parameter *route_obj_name* (or the DB2ROUTE registry value).

The sequence of events is:

a. Client_3 finds the fully qualified DCE name of VANVMDB in its system database directory.
b. Client_3 obtains the database object for VMDB using the DCE name of the database `/.../CELL_VANCOUVER/subsys/database/VMDB`.
c. From this object, Client_3 finds that VMDB only uses the DB protocol DRDA, which is not the protocol that Client_3 uses.
d. Client_3 then obtains the routing information object using the name defined in the *route_obj_name* configuration parameter or the DB2ROUTE registry value. The client finds the target database information for VMDB.
e. Client_3 reads the database locator object associated with the VMDB target database information and retrieves the communication protocol and sends an SQL CONNECT request to the router.
f. The router then sets up an APPC connection with VMDB.

How Directories Are Searched

If the DCE directory is used in an environment where all the target databases share the same directory path name, no local directories are required on the clients.

This section describes the order in which directories are searched for the following:

- ATTACH Command
- CONNECT Statement

ATTACH Command

Figure 9 on page 338 shows how the directories are searched when a client attaches to a database management system instance called ABC_INST.

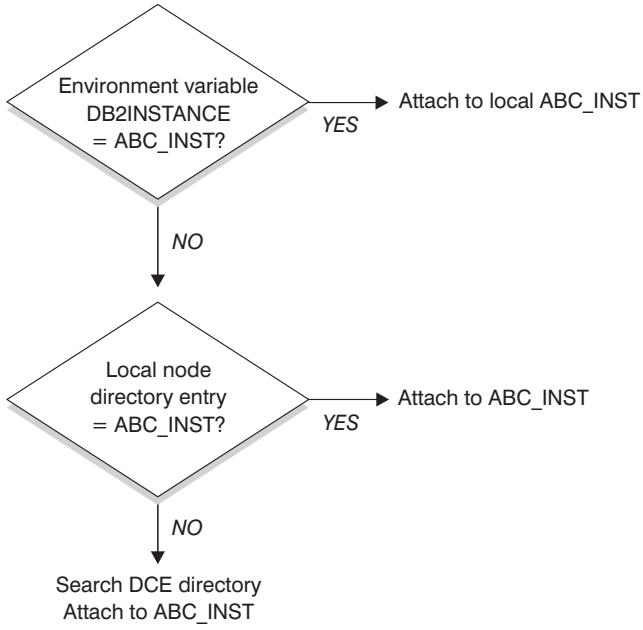


Figure 9. How Directories are Used to Attach a Database

CONNECT Statement

Figure 10 on page 339 shows how the directories are searched when a client connects to a database called DBTEST.

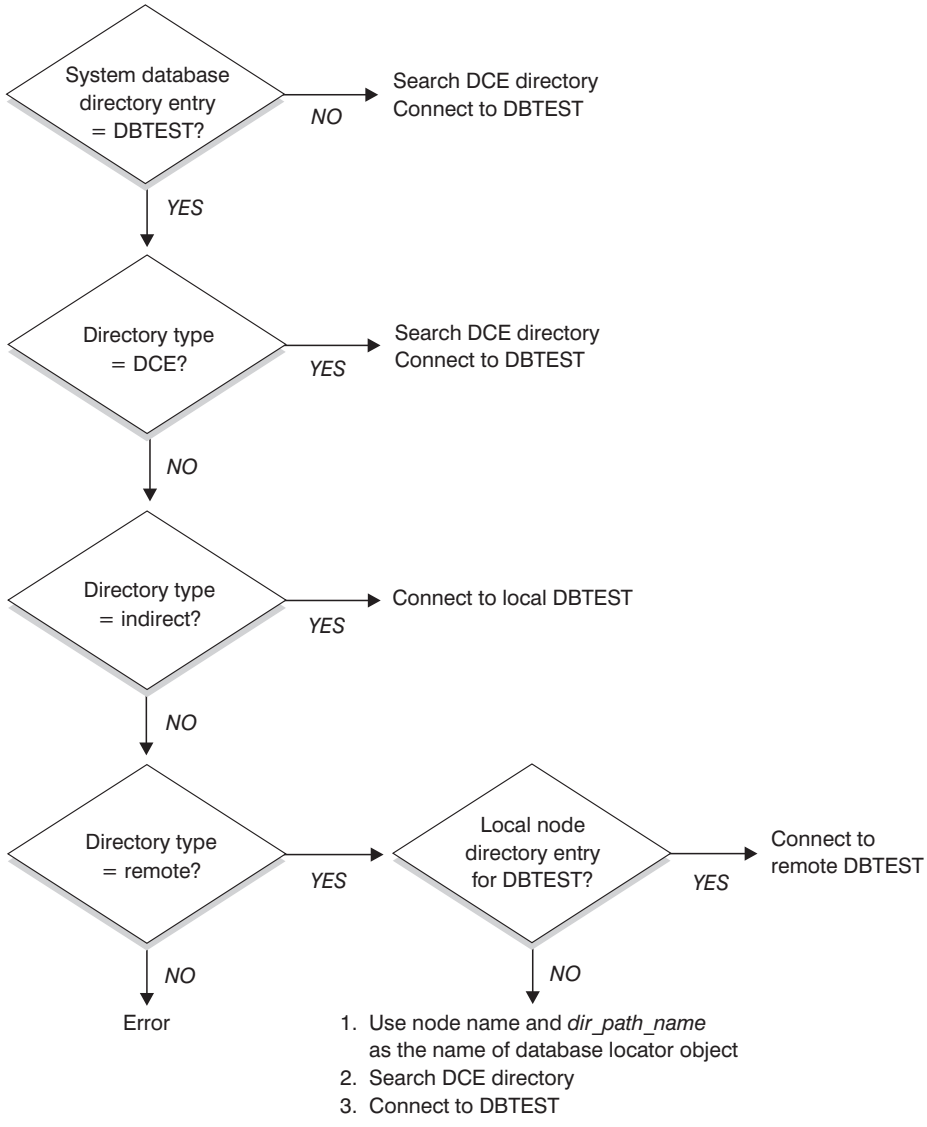


Figure 10. How Directories are Used to Connect a Database

Temporarily Overriding DCE Directory Information

You can use the local database directory to override the DCE directory information. For example, if you CONNECT TO DBTEST where `/.:/subsys/database/DBTEST` is defined in the DCE directory as residing on a host called JAGUAR, you can temporarily change DBTEST to a different

database residing on a host called STORM. Catalog DBTEST locally as a remote database with a node directory entry pointing to STORM.

You can create an alias for a database whose DCE name does not follow the directory path name of the client. See "CATALOG GLOBAL DATABASE Command" on page 332 for details about the command.

Directory Services Tasks

The tasks that must be performed to set up and use DCE Directory Services are listed below. The following sections describe the details of each task.

- DCE Administrator Tasks

  The DCE administrator must update the DCE directory so that the new database resource information can be added.
- Database Administrator Tasks

  The database administrator must update the DCE directory and supply information for DB2 installation and configuration.
- Database User Tasks

  The database user must log in to DCE and know the target database name.

In addition, the network administrator sets up the network access for each user node. Refer to the network documentation for the details.

DCE Administrator Tasks

The DCE administrator must do the following tasks before the directory objects can be created or read:

- Assign the directory subtree for DB2, for example /.:/subsys/database
- Grant the privileges to the database administrator to create directory objects
- Grant the privileges to the database users to read the directory objects
- Add the information for the new DCE directory object attributes to the *DCE attribute table*.

  Edit the CDS attributes file (on UNIX platforms */etc/dce/cds_attributes*; on OS/2 *X:\opt\dcelocal\etc\cds_attr*, where "X" is the appropriate drive) and append the following:

```
1.3.18.0.2.4.30 DB_Comment                char
1.3.18.0.2.4.31 DB_Communication_Protocol char
1.3.18.0.2.4.32 DB_Database_Protocol      char
1.3.18.0.2.4.33 DB_Database_Locator_Name  char
1.3.18.0.2.4.34 DB_Native_Database_Name   char
1.3.18.0.2.4.35 DB_Object_Type            char
1.3.18.0.2.4.36 DB_Product_Name           char
1.3.18.0.2.4.37 DB_Product_Release        char
1.3.18.0.2.4.38 DB_Target_Database_Info   char
1.3.18.0.2.4.39 DB_Authentication         char
1.3.18.0.2.4.63 DB_Principal              char
```

- Ensure DCE is running when users need access to the databases using DCE Directory Services.

For more information, refer to the DCE documentation for the platform you are using.

Database Administrator Tasks

The database administrator must do the following tasks:

- Obtain the directory subtree for the database resources from the DCE administrator. For example, `/.:/subsys/database`
- During installation of the DB2 database manager, ask the DCE administrator to add the new DCE directory object attributes required by DB2.
- Assign a unique name for each database management system instance in the DCE directory subtree. For example, `/.:/subsys/database/AIX_INST`
- For each database management system instance specify the database manager configuration parameters for DCE.
  - *dir_type*
  - *dir_obj_name*
  - *dir_path_name*
  - *route_obj_name*
  - *dft_client_comm*
  - *dft_client_adpt*

  Some of the configuration parameters can be temporarily overridden by registry variables set by the client. Refer to "Configuration Parameters and Registry Variables" on page 331 for more information.
- Assign a unique name for each database in the DCE directory subtree. Specify the name in the *dir_obj_name* parameter in the database configuration file.
- Create the objects for DCE Directory Services using the DCE *cdscp* commands to create and display objects. The objects are created separately from the database manager installation process and the database manager instance start process.

  Three types of objects exist.
  - A database object is required for each target database.
  - A database locator object is required for each DB2 Connect instance and each database management system instance (without DB2 Connect) which is associated with more than one database.
  - Routing information objects are required to access a host database.
- Depending on each environment, the database administrator must determine:

- How to group the clients into logical groups considering what databases they access, and what communications protocols they use.
- How many routing information objects are required.
- Which target databases should be recorded in each routing information object.
- Which routing information objects should be known to which group of clients.

Refer to "Creating Directory Objects" on page 319 for details about the objects.

Database User Tasks

The database user must do the following tasks:

- Obtain the name of the database from the database administrator. This name can be a simple one-part name, or a fully-qualified DCE name.
- If needed, specify the values required for DCE Directory Services in the registry variables. Registry variables set by the client can temporarily override the configuration parameters.
  - If host database access is required, obtain the fully-qualified DCE name of the routing information object from the database administrator. If this name is not specified in the *route_obj_name*, or it is a different name, specify this name in the DB2ROUTE registry variable before trying to connect to the host database.
  - If your preferred communication protocol is not specified in *dft_client_comm*, or it is a different protocol, specify the communication protocol for the client in the DB2CLIENTCOMM registry variable. Here are **some** UNIX examples:

    ```
    db2set DB2CLIENTCOMM=tcpip
    db2set DB2CLIENTCOMM=appc
    db2set DB2CLIENTCOMM=tcpip,appc
    db2set DB2CLIENTCOMM=appc,tcpip
    ```

    Some OS/2 examples are:

    ```
    db2set DB2CLIENTCOMM=ipxspx
    db2set DB2CLIENTCOMM=netbios
    db2set DB2CLIENTCOMM=tcpip,ipxspx,netbios
    db2set DB2CLIENTCOMM=netbios,tcpip,ipxspx,appc
    ```

    Some Windows operating system examples are:

    ```
    db2set DB2CLIENTCOMM=npipe
    db2set DB2CLIENTCOMM=netbios
    db2set DB2CLIENTCOMM=tcpip,ipxspx,netbios
    db2set DB2CLIENTCOMM=netbios,tcpip,ipxspx,appc,npipe
    ```

If more than one communication protocol exists, the first one specified is used.

- If any of the databases has a DCE name that is not in the directory path defined in the *dir_path_name* configuration parameter or the DB2DIRPATHNAME registry variable, then explicitly catalog the database with the CATALOG GLOBAL DATABASE command. Refer to "CATALOG GLOBAL DATABASE Command" on page 332 for more information.
- Log in to DCE before connecting to the target database or attaching to the database instance. Refer to the *OSF DCE Administration Guide* for more information about the login command.

Directory Services Restrictions

This section describes what is not supported.

- Not all database clients may be supported. See your *Quick Beginnings* manual to determine whether DCE directory services is supported from your DB2 client. Currently, support is only provided for DB2 Clients for all UNIX, OS/2 and supported Windows operating systems.
- A client cannot use DCE Directory Services to connect to a DB2 for OS/2 Version 1 server.
- Only supported Windows operating systems clients can use any or all of the TCP/IP, APPC, NetBIOS, IPX/SPX, or NPIPE protocols. Only OS/2 clients can use any or all of the TCP/IP, APPC, NetBIOS, and IPX/SPX protocols. All supported UNIX clients can only use the TCP/IP and APPC protocols.
- LIST DATABASE (or NODE) DIRECTORY COMMANDS only provide entries from the local directories and not entries from the DCE directory. You can use the *cdscp show object* command in DCE to display the objects.
- When all of the following conditions exist, the owner of the database manager instance must log in to DCE before starting the database manager (using the `db2start` command).
  - The database manager instance is configured to support DCE directory services through the *dir_type* configuration parameter
  - The cell directory services object can only be read by explicitly logging into DCE
  - The DCE directory must be accessed to support either of the following:
    - A transaction manager database (specified by the *tm_database* configuration parameter) located on another instance
    - A client that cannot support DCE directory services, or is not configured to use DCE directory services.

Note: When performing the DCE login, you should use a principal that has a long ticket lifetime.

- When using a DDCS Version 2.2 (or earlier) gateway to connect a client that is using DCE directory services to a DRDA server, you must catalog the database alias in the gateway's local directory. This database alias must be the same as the alias on the client and it must represent the same database.
- When using supported Windows operating systems clients, DB2DCE.DLL will be used. This file is found in the *bin* subdirectory of the *sqllib* subdirectory. If the DCE provider is Gradient**, by default the file DB2DCE.GRD is equivalent to DB2DCE.DLL. If the DCE provider is IBM, the file DB2DCE.IBM must be copied to DB2DCE.DLL.

Appendix C. User Exit for Database Recovery

Provide overview information here similar to that found for the Data Movement chapter.

Note: All of the information on these topics, and the comparable topics from the *Command Reference* and the *Administrative API Reference*, have been consolidated into the *Data Recovery and High Availability Guide and Reference*.

The *Data Recovery and High Availability Guide and Reference* is your primary, single source of information for these topics.

Appendix D. Issuing Commands to Multiple Database Partitions

In a partitioned database system, you may want to issue commands to be run on machines in the instance, or on database partition servers (nodes). You can do so using the **rah** command or the **db2_all** command. The **rah** command allows you to issue commands that you want to run at machines in the instance. If you want the commands to run at database partition servers in the instance, you run the **db2_all** command. This section provides an overview of these commands. The information that follows applies to partitioned database systems only.

Notes:

1. On UNIX-based platforms, your login shell can be a Korn shell or any other shell; however, there are differences in the way the different shells handle commands containing special characters.
2. On Windows NT, to run the **rah** command or the **db2_all** command, you must be logged on with a user account that is a member of the Administrators group.

To determine the scope of a command, refer to the *Command Reference*. This book indicates whether a command runs on a single database partition server, or on all of them. If the command runs on one database partition server and you want it to run on all of them, use **db2_all**. The exception is the **db2trc** command, which runs on all the logical nodes (database partition servers) on a machine. If you want to run **db2trc** on all logical nodes on all machines, use **rah**.

Commands

You can run the commands sequentially at one database partition server after another, or you can run the commands in parallel. On UNIX-based platforms, if you run the commands in parallel, you can either choose to have the output sent to a buffer and collected for display (the default behavior) or the output can be displayed at the machine where the command is issued. On Windows NT, if you run the commands in parallel, the output is displayed at the machine where the command is issued.

To use the **rah** command, type:

```
rah command
```

To use the **db2_all** command, type:

```
db2_all command
```

To obtain help about **rah** syntax, type

```
rah "?"
```

The command can be almost anything which you could type at an interactive prompt, including, for example, multiple commands to be run in sequence. On UNIX-based platforms, you separate multiple commands using a semicolon (;). On Windows NT, you separate multiple commands using an ampersand (&). Do not use the separator character following the last command.

The following example shows how to use the **db2_all** command to change the database configuration on all database partitions that are specified in the node configuration file. Because the ; character is placed inside double quotation marks, the request will run concurrently:

```
db2_all ";UPDATE DB CFG FOR sample USING LOGFILSIZ=100"
```

Command Descriptions

You can use the following commands:

Command	Description
rah	Runs the command on all machines.
db2_all	Runs the command on all database partition servers that you specify.
db2_kill	Abruptly stops all processes being run on multiple database partition servers and cleans up all resources on all database partition servers. This command renders your databases inconsistent. Do *not* issue this command except under direction from IBM service.
db2_call_stack	On UNIX-based platforms, causes all processes running on all database partition servers to write call traceback to the syslog. On Windows NT, causes all processes running on all database partition servers to write call traceback to the P*xxxx*.*nnn* file in the instance directory, where P*xxxx* is the process ID and *nnn* is the node number.

On UNIX-based platforms, these commands execute **rah** with certain implicit settings such as:

- Run in parallel at all machines
- Buffer command output in `/tmp/$USER/db2_kill`, `/tmp/$USER/db2_call_stack` respectively.

On Windows NT, these commands execute **rah** to run in parallel at all machines.

Specifying the Command to Run

You can specify the command:

- From the command line as the parameter
- In response to the prompt if you don't specify any parameter.

You should use the prompt method if the command contains the following special characters:

```
| & ; < > ( )  { } [ ] unsubstituted $
```

If you specify the command as the parameter on the command line, you must enclose it in double quotation marks if it contains any of the special characters just listed.

Note: On UNIX-based platforms, the command will be added to your command history just as if you typed it at the prompt.

All special characters in the command can be entered normally (without being enclosed in quotation marks, except for \). If you need to include a \ in your command, you must type two backslashes (\\).

Note: On UNIX-based platforms, if you are not using a Korn shell, all special characters in the command can be entered normally (without being enclosed in quotation marks, except for ", \, unsubstituted $, and the single quotation mark (')). If you need to include one of these characters in your command, you must precede them by three backslashes (\\\). For example, if you need to include a \ in your command, you must type four backslashes (\\\\).

If you need to include a double quotation mark (") in your command, you must precede it by three backslashes, for example, \\\".

Notes:

1. On UNIX-based platforms, you cannot include a single quotation mark (') in your command unless your command shell provides some way of entering a single quotation mark inside a singly quoted string.
2. On Windows NT, you cannot include a single quotation mark (') in your command unless your command window provides some way of entering a single quotation mark inside a singly quoted string.

When you run any korn-shell shell-script which contains logic to read from stdin in the background, you should explicitly redirect stdin to a source where the process can read without getting stopped on the terminal (SIGTTIN message). To redirect stdin, you can run a script with the following form:

```
shell_script </dev/null &
```

if there is no input to be supplied.

In a similar way, you should always specify `</dev/null` when running db2_all in the background. For example:

```
db2_all ";run_this_command" </dev/null &
```

By doing this you can redirect stdin and avoid getting stopped on the terminal.

An alternative to this method, when you are not concerned about output from the remote command, is to use the "daemonize" option in the db2_all prefix:

```
db2_all ";daemonize_this_command" &
```

Running Commands in Parallel on UNIX-Based Platforms

Note: The information in this section applies to UNIX-based platforms only.

By default, the command is run sequentially at each machine, but you can specify to run the commands in parallel using background rshells by prefixing the command with certain prefix sequences. If the rshell is run in the background, then each command puts the output in a buffer file at its remote machine. This process retrieves the output in two pieces:

1. After the remote command completes.
2. After the rshell terminates, which may be later if some processes are still running.

The name of the buffer file is `/tmp/$USER/rahout` by default, but it can be specified by the environment variables $RAHBUFDIR/$RAHBUFNAME.

When you specify that you want the commands to be run concurrently, by default, this script prefixes an additional command to the command sent to all hosts to check that $RAHBUFDIR and $RAHBUFNAME are usable for the buffer file. It creates $RAHBUFDIR. To suppress this, export an environment variable `RAHCHECKBUF=no`. You can do this to save time if you know the directory exists and is usable.

Before using **rah** to run a command concurrently at multiple machines:

- Ensure that a directory `/tmp/$USER` exists for your user ID at each machine. To create a directory if one does not already exist, run:

  ```
  rah ")mkdir /tmp/$USER"
  ```

- Add the following line to your `.kshrc` (for Korn shell syntax) or `.profile`, and also type it into your current session:

  ```
  export RAHCHECKBUF=no
  ```

- Ensure that each machine ID at which you run the remote command has an entry in its `.rhosts` file for the ID which runs **rah**; and the ID which runs **rah** has an entry in its `.rhosts` file for each machine ID at which you run the remote command.

Monitoring rah Processes on UNIX-Based Platforms

Note: The information in this section applies to UNIX-based platforms only. While any remote commands are still running or buffered output is still being accumulated, processes started by `rah` monitor activity to:

- Write messages to the terminal indicating which commands have not been run
- Retrieve buffered output.

The informative messages are written at an interval controlled by the environment variable RAHWAITTIME. Refer to the help information for details on how specify this. All informative messages can be completely suppressed by exporting `RAHWAITTIME=0`.

The primary monitoring process is a command whose command name (as shown by the ps command) is **rahwaitfor**. The first informative message tells you the pid (process id) of this process. All other monitoring processes will appear as **ksh** commands running the **rah** script (or the name of the symbolic link). If you want, you can stop all monitoring processes by the command:

```
kill <pid>
```

where <pid> is the process ID of the primary monitoring process. Do not specify a signal number. Leave the default of 15. This will not affect the remote commands at all, but will prevent the automatic display of buffered output. Note that there may be two or more different sets of monitoring processes executing at different times during the life of a single execution of **rah**. However, if at any time you stop the current set, then no more will be started.

If your regular login shell is not a Korn shell (for example `/bin/ksh`), you can use **rah**, but there are some slightly different rules on how to enter commands containing the following special characters:

```
"  unsubstituted $ '
```

For more information, type `rah "?"`. Also, in a UNIX-based environment, if the login shell at the ID which executes the remote commands is not a Korn shell, then the login shell at the ID which executes **rah** must also not be a Korn shell. (**rah** makes the decision as to whether the remote ID's shell is a Korn shell based on the local ID). The shell must not perform any substitution or special processing on a string enclosed in single quotation marks. It must leave it exactly as is.

Additional rah (Run All Hosts) Information (Solaris and AIX Only)

To enhance performance, rah has been extended to use tree_logic on large systems. That is, rah will check how many nodes the list contains, and if that number exceeds a threshold value, it constructs a subset of the list and sends a recursive invocation of itself to those nodes. At those nodes, the recursively invoked rah follows the same logic until the list is small enough to follow the standard logic (now the ″leaf-of-tree″ logic) of sending the command to all nodes on the list. The threshold can be specified by environment variable RAHTREETHRESH, or defaults to 15.

In the case of a multiple-logical-node-per-physical-node system, db2_all will favor sending the recursive invocation to distinct physical nodes, which will then rsh to other logical nodes on the same physical node, thus also reducing inter-physical-node traffic. (This point applies only to db2_all, not rah, since rah always sends only to distinct physical nodes.)

Prefix Sequences

A prefix sequence is one or more special characters. Type one or more prefix sequences immediately preceding the characters of the command without any intervening blanks. If you want to specify more than one sequence, you can type them in any order, but characters within any multicharacter sequence must be typed in order. If you type any prefix sequences, you must enclose the entire command, including the prefix sequences in double quotation marks, as in the following examples:

- On UNIX-based platforms:

```
rah "};ps -F pid,ppid,etime,args -u $USER"
```

- On Windows NT:

```
rah "||db2 get db cfg for sample"
```

The prefix sequences are:

Sequence	Purpose
|	Runs the commands in sequence in the background.
|&	Runs the commands in sequence in the background and terminates the command after all remote commands have completed, even if some processes are still running. This may be later if, for example, child processes (on UNIX-based platforms) or background processes (on Windows NT) are still running. In this case, the command starts a separate background process to retrieve any remote output generated after command termination and writes it back to the originating machine.

		Note: On UNIX-based platforms, specifying & degrades performance, because more **rsh** commands are required.
	||	Runs the commands in parallel in the background.
	||&	Runs the commands in parallel in the background and terminates the command after all remote commands have completed as described for the |& case above. **Note:** On UNIX-based platforms, specifying & degrades performance, because more **rsh** commands are required.
	;	Same as ||& above. This is an alternative shorter form. **Note:** On UNIX-based platforms, specifying ; degrades performance relative to ||, because more **rsh** commands are required.
	]	Prepends dot-execution of user's profile before executing command. **Note:** Available on UNIX-based platforms only.
	}	Prepends dot-execution of file named in $RAHENV (probably `.kshrc`) before executing command. **Note:** Available on UNIX-based platforms only.
	]}	Prepends dot-execution of user's profile followed by execution of file named in $RAHENV (probably `.kshrc`) before executing command. **Note:** Available on UNIX-based platforms only.
	)	Suppresses execution of user's profile and of file named in $RAHENV. **Note:** Available on UNIX-based platforms only.
	'	Echoes the command invocation to the machine.
	<	Sends to all the machines except this one.
	<<–nnn<	Sends to all-but-database partition server *nnn* (all database partition servers in `db2nodes.cfg` except for node number *nnn*, see the first paragraph following the last prefix sequence in this table).
	<<+nnn<	Sends to only database partition server *nnn* (the database

partition server in db2nodes.cfg whose node number is *nnn*, see the first paragraph following the last prefix sequence in this table).

(blank character)
Runs the remote command in the background with stdin, stdout, and stderr all closed. This option is valid only when running the command in the background, that is, only in a prefix sequence which also includes \ or ;. It allows the command to complete much sooner (as soon as the remote command has been initiated). If you specify this prefix sequence on the **rah** command line, then either enclose the command in single quotation marks, or enclose the command in double quotation marks, and precede the prefix character by \ . For example,

```
rah '; mydaemon'
```

or

```
rah ";\ mydaemon"
```

When run as a background process, the **rah** command will never wait for any output to be returned.

>
Substitutes occurrences of <> with the machine name.

"
Substitutes occurrences of () by the machine index, and substitutes occurrences of ## by the node number.

Notes:

1. The machine index is a number that associated with a machine in the database system. If you are not running multiple logical nodes, the machine index for a machine corresponds to the node number for that machine in the node configuration file. To obtain the machine index for a machine in a multiple logical node environment, do not count duplicate entries for those machines that run multiple logical nodes. For example, if MACH1 is running two logical nodes and MACH2 is also running two logical nodes, the node number for MACH3 is 5 in the node configuration file. The machine index for MACH3, however, would be 3.

   On Windows NT, do not edit the node configuration file. To obtain the machine index, use the **db2nlist** command. Refer to the *DB2 Enterprise - Extended Edition for Windows Quick Beginnings* manual for details.

2. When " is specified, duplicates are not eliminated from the list of machines. See "Eliminating Duplicate Entries from the List of Machines" if you want to eliminate duplicates.

When using the <<-nnn< and <<+nnn< prefix sequences, *nnn* is any 1-, 2- or 3-digit partition number which must match the *nodenum* value in the db2nodes.cfg file.

Note: Prefix sequences are considered to be part of the command. If you specify a prefix sequence as part of a command, you must enclose the entire command, including the prefix sequences, in double quotation marks.

Specifying the List of Machines

By default, the list of machines is taken from the node configuration file, db2nodes.cfg. You can override this by:

- Specifying a pathname to the file that contains the list of machines by exporting (on UNIX-based platforms) or setting (on Windows NT) the environment variable RAHOSTFILE.
- Specifying the list explicitly, as a string of names separated by spaces, by exporting (on UNIX-based platforms) or setting (on Windows NT) the environment variable RAHOSTLIST.

  Note: If both of these environment variables are specified, RAHOSTLIST takes precedence.

Note: On Windows NT, to avoid introducing inconsistencies into the node configuration file, do *not* edit it manually. To obtain the list of machines in the instance, use the **db2nlist** command. Refer to the *DB2 Enterprise - Extended Edition for Windows Quick Beginnings* manual for details.

Eliminating Duplicate Entries from the List of Machines

If you are running DB2 Enterprise - Extended Edition with multiple logical nodes (database partition servers) on one machine, your db2nodes.cfg file will contain multiple entries for that machine. In this situation, the **rah** command needs to know whether you want the command to be executed once only on each machine or once for each logical node listed in the db2nodes.cfg file. Use the **rah** command to specify machines. Use the **db2_all** command to specify logical nodes.

Note: On UNIX-based platforms, if you specify machines, **rah** will normally eliminate duplicates from the machine list, with the following exception: if you specify logical nodes, **db2_all** prepends the following assignment to your command:

```
export DB2NODE=nnn  (for Korn shell syntax)
```

where *nnn* is the node number taken from the corresponding line in the `db2nodes.cfg` file, so that the command will be routed to the desired database partition server.

When specifying logical nodes, you can restrict the list to include all logical nodes except one, or only specify one database partition server using the `<<-nnn<` and `<<+nnn<` prefix sequences. You may want to do this if you want to run a command at the catalog node first, and when that has completed, run the same command at all other database partition servers, possibly in parallel. This is usually required when running the **db2 restart database** command. You will need to know the node number of the catalog node to do this. See "Prefix Sequences" on page 352 for information about the prefix sequences.

If you execute **db2 restart database** using the **rah** command, duplicate entries are eliminated from the list of machines. However if you specify the " prefix, then duplicates are not eliminated, because it is assumed that use of the " prefix implies sending to each database partition server, rather than to each machine.

Controlling the rah Command

You can use the following environment variables to control the **rah** command.

Table 23.

Name	Meaning	Default
$RAHBUFDIR **Note:** Available on UNIX-based platforms only.	Directory for buffer	/tmp/$USER
$RAHBUFNAME **Note:** Available on UNIX-based platforms only.	Filename for buffer	rahout
$RAHOSTFILE (on UNIX-based platforms); RAHOSTFILE (on Windows NT)	File containing list of hosts	`db2nodes.cfg`
$RAHOSTLIST (on UNIX-based platforms); RAHOSTLIST (on Windows NT)	List of hosts as a string	extracted from $RAHOSTFILE

Table 23. (continued)

Name	Meaning	Default
$RAHCHECKBUF **Note:** Available on UNIX-based platforms only.	If set to ″no″, bypass checks	not set
$RAHSLEEPTIME (on UNIX-based platforms); RAHSLEEPTIME (on Windows NT)	Time in seconds this script will wait for initial output from commands run in parallel	86400 seconds for **db2_kill**, 200 seconds for all other
$RAHWAITTIME (on UNIX-based platforms); RAHWAITTIME (on Windows NT)	On Windows NT, interval in seconds between successive checks that remote jobs are still running. On UNIX-based platforms, interval in seconds between successive checks that remote jobs are still running and `rah: waiting for <pid> ...` messages. On all platforms, specify any positive integer. Prefix value with a leading zero to suppress messages, for example, export RAHWAITTIME=045. It is not necessary to specify a low value as **rah** does not rely on these checks to detect job completion.	45 seconds
$RAHENV **Note:** Available on UNIX-based platforms only.	Specifies filename to be executed if $RAHDOTFILES=E or K or PE or B	$ENV
$RAHUSER (on UNIX-based platforms); RAHUSER (on Windows NT)	On UNIX-based platforms, user ID under which the remote command is to be run. On Windows NT, the logon account associated with the DB2 Remote Command Service	$USER

Note: On UNIX-based platforms, the value of $RAHENV where **rah** is run is used, not the value (if any) set by the remote shell.

$RAHDOTFILES on UNIX-Based Platforms

Note: The information in this section applies to UNIX-based platforms only. Following are the ·files that are run if no prefix sequence is specified:

P .profile

E File named in $RAHENV (probably `.kshrc`)

K Same as E

PE .profile followed by file named in $RAHENV (probably `.kshrc`)

B Same as PE

N None (or Neither)

Note: If your login shell is not a Korn shell, any dot files you specify to be executed will be executed in a Korn shell process, and so must conform to Korn shell syntax. So, for example, if your login shell is a C shell, to have your `.cshrc` environment set up for commands executed by **rah**, you should either create a Korn shell `INSTHOME/.profile` equivalent to your `.cshrc` and specify in your `INSTHOME/.cshrc`:

```
setenv RAHDOTFILES P
```

or you should create a Korn shell `INSTHOME/.kshrc` equivalent to your `.cshrc` and specify in your `INSTHOME/.cshrc`:

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

Also, it is essential that your `.cshrc` does not write to stdout if there is no tty (as when invoked by **rsh**). You can ensure this by enclosing any lines which write to stdout by, for example,

```
if { tty -s } then echo "executed .cshrc";
endif
```

Setting the Default Environment Profile on Windows NT

Note: The information in this section applies to Windows NT only.

To set the default environment profile for the **rah** command, use a file called `db2rah.env`, which should be created in the instance directory. The file should have the following format:

```
; This is a comment line
DB2INSTANCE=instancename
DB2DBDFT=database
; End of file
```

You can specify all the environment variables that you need to initialize the environment for **rah**.

Determining Problems with rah on UNIX-Based Platforms

Note: The information in this section applies to UNIX-based platforms only. Here are suggestions on how to handle some problems that you may encounter when you are running **rah**:

1. **rah** hangs (or takes a very long time)

   This problem may be caused because:

   - **rah** has determined that it needs to buffer output, and you did not export `RAHCHECKBUF=no`. Therefore, before running your command, **rah** sends a command to all machines to check the existence of the buffer directory, and to create it if it does not exist.
   - One or more of the machines where you are sending your command is not responding. The **rsh** command will eventually time out but the time-out interval is quite long, usually about 60 seconds.

2. You have received messages such as:

   - Login incorrect
   - Permission denied

   Either one of the machines does not have the ID running **rah** correctly defined in its `.hosts` file, or the ID running **rah** does not have one of the machines correctly defined in its `.rhosts` file.

3. When running commands in parallel using background rshells, although the commands run and complete within the expected elapsed time at the machines, **rah** takes a long time to detect this and put up the shell prompt.

   The ID running **rah** does not have one of the machines correctly defined in its `.rhosts` file.

4. Although **rah** runs fine when run from the shell command line, if you run **rah** remotely using rsh, for example,

   ```
   rsh somewher -l $USER db2_kill
   ```

   rah never completes.

   This is normal. **rah** starts background monitoring processes, which continue to run after it has exited. Those processes will normally persist until all processes associated with the command you ran have themselves terminated. In the case of **db2_kill**, this means termination of all database managers. You can terminate the monitoring processes by finding the process whose command is **rahwaitfor** and kill <process_id>. Do not specify a signal number. Instead, use the default (15).

5. The output from **rah** is not displayed correctly, or **rah** incorrectly reports that $RAHBUFNAME does not exist, when multiple commands of **rah** were issued under the same $RAHUSER.

This is because multiple concurrent executions of **rah** are trying to use the same buffer file (for example, $RAHBUFDIR/$RAHBUFNAME) for buffering the outputs. To prevent this problem, use a different $RAHBUFNAME for each concurrent **rah** command, for example in the following ksh:

```
export RAHBUFNAME=rahout
rah ";$command_1" &
export RAHBUFNAME=rah2out
rah ";$command_2" &
```

or use a method that makes the shell choose a unique name automatically such as:

```
RAHBUFNAME=rahout.$$ db2_all "....."
```

Whatever method you use, you must ensure you clean up the buffer files at some point if disk space is limited. **rah** does not erase a buffer file at the end of execution, although it will erase and then re-use an existing file the next time you specify the same buffer file.

6. You entered

   ```
   rah '"print from ()'
   ```

   and received the message:

   ```
   ksh: syntax error at line 1 : (' unexpected
   ```

   Prerequisites for the substitution of () and ## are:

   - Use **db2_all**, not **rah**.
   - Ensure a RAHOSTFILE is used either by exporting RAHOSTFILE or by defaulting to your `/sqllib/db2nodes.cfg` file. Without these prerequisites, **rah** will leave the () and ## as is. You receive an error because the command **print from ()** is not valid.

   For a performance tip when running commands in parallel, use `|` rather than `|&`, and use `||` rather than `||&` or `;` unless you truly need the function provided by &. Specifying & requires more **rsh** commands and therefore degrades performance.

Appendix E. How DB2 for Windows NT Works with Windows NT Security

When you install Windows NT, it allows you to create two administrator usernames:

- One is called "Administrator"
- The other is a name of your choice. It must have administrator authority and must comply with DB2's naming rules. For more information on DB2's naming rules, see "Appendix A. Naming Rules" on page 313.

The user may log on to the local machine, or when the machine is installed in a Windows NT Domain, the user may log on to the Domain. DB2 for Windows NT supports both of these options. To authenticate the user, DB2 checks the local machine first, then the Domain Controller for the current Domain, and finally any Trusted Domains known to the Domain Controller.

To illustrate how this works, suppose that the DB2 instance requires Server authentication. The configuration is as follows:

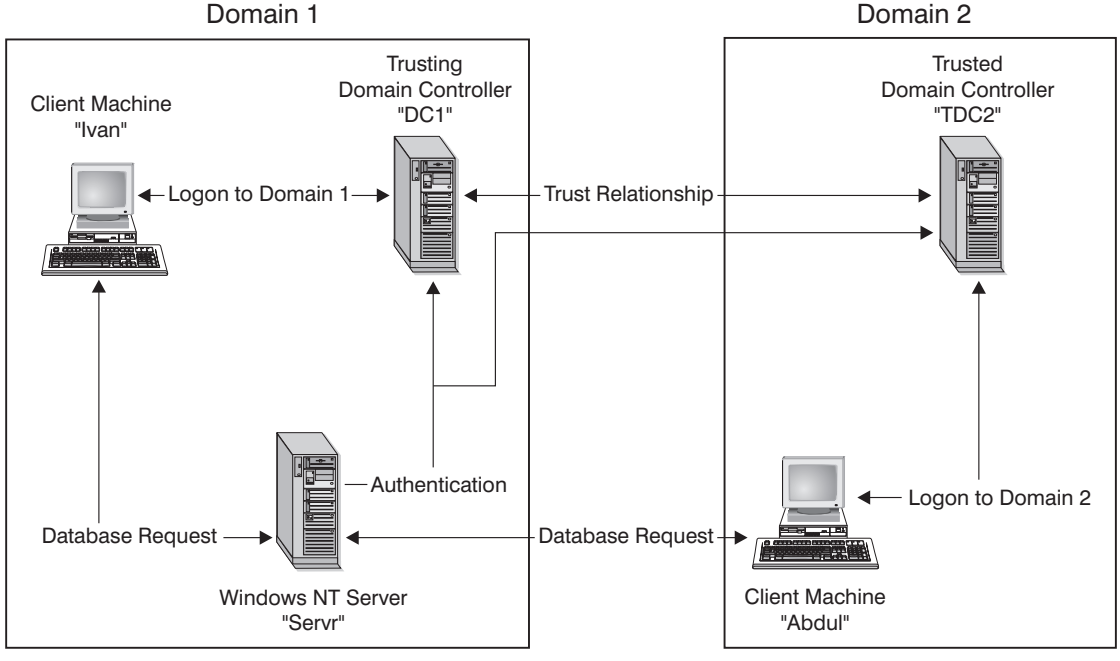


Figure 11. Authentication Using Windows NT Domains

Each machine has a security database, Security Access Management (SAM), unless a client machine is running Windows 9x. Windows 9x machines do not have a SAM database. DC1 is the domain controller, in which the client machine, Ivan, and the DB2 for Windows NT server, Servr, are enrolled. TDC2 is a trusted domain for DC1 and the client machine, Abdul, is a member of TDC2's domain.

A Sample Scenario with Server Authentication:

1. Abdul logs on to the TDC2 domain (that is, he is known in the TDC2 SAM database).
2. Abdul then connects to a DB2 database that is cataloged to reside on SRV3:

   ```
   db2 connect to remotedb user Abdul using fredpw
   ```

3. SRV3 determines where Abdul is known. The API that is used to find this information first searches the local machine (SRV3) and then the domain controller (DC1) before trying any trusted domains. Username Abdul is found on TDC2. This search order requires a single namespace for users and groups.
4. SRV3 then:
   a. Validates the username and password with TDC2.
   b. Finds out whether Abdul is an administrator by asking TDC2.
   c. Enumerates all Abdul's groups by asking TDC2.

A Sample Scenario with Client Authentication and a Windows NT Client Machine:

1. Dale, the administrator, logs on to SRV3 and changes the authentication for the database instance to Client:

   ```
   db2 update dbm cfg using authentication client
   db2stop myinst
   db2start myinst
   ```

2. Ivan, at a Windows client machine, logs on to the DC1 domain (that is, he is known in the DC1 SAM database).
3. Ivan then connects to a DB2 database that is cataloged to reside on SRV3:

   ```
   DB2 CONNECT to remotedb user Ivan using johnpw
   ```

4. Ivan's machine validates the username and password. The API used to find this information first searches the local machine (Ivan) and then the domain controller (DC1) before trying any trusted domains. Username Ivan is found on DC1.
5. Ivan's machine then validates the username and password with DC1.
6. SRV3 then:
   a. Determines where Ivan is known.
   b. Finds out whether Ivan is an administrator by asking DC1.

c. Enumerates all Ivan's groups by asking DC1.

Note: Before attempting to connect to the DB2 database, ensure that DB2 Security Service has been started. The Security Service is installed by DB2 and is set up to run as a Windows NT service; however, it is not started automatically. To start the DB2 Security Service, enter the `NET START DB2NTSECSERVER` command.

A Sample Scenario with Client Authentication and a Windows 95 Client Machine:

1. Dale, the administrator, logs on to SRV3 and changes the authentication for the database instance to Client:

   ```
   db2 update dbm cfg using authentication client
   db2stop myinst
   db2start myinst
   ```

2. Ivan, at a Windows 95 client machine, logs on to the DC1 domain (that is, he is known in the DC1 SAM database).
3. Ivan then connects to a DB2 database that is cataloged to reside on SRV3:

   ```
   db2 connect to remotedb user Ivan using johnpw
   ```

4. Ivan's Windows 95 machine cannot validate the username and password. The username and password are therefore assumed to be valid.
5. SRV3 then:
   a. Determines where Ivan is known.
   b. Finds out whether Ivan is an administrator by asking DC1.
   c. Enumerates all Ivan's groups by asking DC1.

Note: Because a Windows 95 client cannot validate a given username and password, client authentication under Windows 95 is inherently insecure. If the Windows 95 machine has access to a Windows NT security provider, however, some measure of security can be imposed by configuring the Windows 95 system for validated pass-through logon. For details on how to configure your Windows 95 system in this way, refer to the Microsoft documentation for Windows 95.

DB2 also supports global groups. In order to use global groups, you must include global groups inside a local group that is on the security server. When DB2 enumerates all the groups that a person is a member of, it also lists the local groups the user is a member of indirectly (by the virtue of being in a global group that is itself a member of one or more local groups).

Using a Backup Domain Controller with DB2

If the server you use for DB2 also acts as a backup domain controller, you can improve DB2 performance and reduce network traffic if you configure DB2 to use the backup domain controller.

You specify the backup domain controller to DB2 by setting the *DB2DMNBCKCTLR* registry variable.

If you know the name of the domain for which DB2 server is the backup domain controller, use:

```
db2dmnbckctlr=DOMAIN_NAME
```

where *DOMAIN_NAME* must be in upper case.

To have DB2 determine the domain for which the local machine is a backup domain controller, use:

```
DB2DMNBCKCTLR=?
```

Note: DB2 does not use an existing backup domain controller by default because a backup domain controller can get out-of-sync with the primary domain controller, causing a security exposure. Domain controllers get out-of-sync when the primary domain controller's security database is updated but the changes are not propagated to a backup domain controller. This can happen if there are network latencies or if the computer browser service is not operational.

User Authentication with DB2 for Windows NT

User authentication can cause problems for Windows NT users because of the way the operating system authenticates. This section describes some considerations for user authentication under DB2 for Windows NT:

- "User Name and Group Name Restrictions"
- "DB2 for Windows NT Security Service" on page 365
- "Installing DB2 on a Backup Domain Controller" on page 365
- "Authentication With Groups and Domain Security" on page 366

User Name and Group Name Restrictions

The following are the limitations in this environment:

- User names names are limited to 30 characters within DB2. Group names are limited to 8 characters.
- User names under Windows NT are not case sensitive; however, passwords are case sensitive.
- User names and group names can be a combination of upper- and lowercase characters. However, they are usually converted to uppercase when used within DB2. For example, if you connect to the database and create the table schema1.table1, this table is stored as SCHEMA1.TABLE1 within the database. (If you wish to use lowercase object names, issue commands from the command line processor, enclosing the object names in quotation marks, or use third-party ODBC front-end tools.)

DB2 for Windows NT Security Service

In DB2 Universal Database we have integrated the authentication of user names and passwords into the DB2 System Controller. The Security Service is only required when a client is connected to a server that is configured for authentication CLIENT.

Installing DB2 on a Backup Domain Controller

In a Windows NT environment a user can be authenticated at either a primary or a backup controller. This feature is very important in large distributed LANs with one central primary domain controller and one or more backup domain controllers (BDC) at each site. Users can then be authenticated on the backup domain controller at their site instead of requiring a call to the primary domain controller (PDC) for authentication.

The advantage of having a backup domain controller, in this case, is that users are authenticated faster and the LAN is not as congested as it would have been had there been no BDC.

Authentication can occur at the BDC under the following conditions:

- The DB2 for Windows NT server is installed on the backup domain controller.
- The DB2DMNBCKCTLR profile registry variable is set appropriately.

If the DB2DMNBCKCTLR profile registry variable is not set or is set to blank, DB2 for Windows NT performs authentication at the primary domain controller.

The only valid declared settings for DB2DMNBCKCTLR are "?" or a domain name.

If the DB2DMNBCKCTLR profile registry variable is set to a question mark (DB2DMNBCKCTLR=?) then DB2 for Windows NT will perform its authentication on the backup domain controller under the following conditions:

- The cachedPrimaryDomain is a registry value set to the name of the domain to which this machine belongs. (You can find this setting under **HKEY_LOCAL_MACHINE—> Software—> Microsoft—> Windows NT—> Current Version—> WinLogon**.)
- The Server Manager shows the backup domain controller as active and available. (That is, the icon for this machine is not greyed out.)
- The registry for the DB2 Windows NT server indicates that the system is a backup domain controller on the specified domain.

Under normal circumstances the setting DB2DMNBCKCTLR=? will work; however, it will not work in all environments. The information supplied about

the servers on the domain is dynamic, and Computer Browser must be running to keep this information accurate and current. Large LANs may not be running Computer Browser and therefore Server Manager's information may not be current. In this case, there is a second method to tell DB2 for Windows NT to authenticate at the backup domain controller: set DB2DMNBCKCTLR=*xxx* where *xxx* is the Windows NT domain name for the DB2 server. With this setting, authentication will occur on the backup domain controller based on the following conditions:

- The cachedPrimaryDomain is a registry value set to the name of the domain to which this machine belongs. (You can find this setting under **HKEY_LOCAL_MACHINE—> Software—> Microsoft—> Windows NT—> Current Version—> WinLogon**.)
- The machine is configured as a backup domain controller for the specified domain. (If the machine is set up as a backup domain controller for another domain, this setting will result in an error.)

Authentication With Groups and Domain Security

DB2 for Windows NT supports the following types of groups:

- Local groups
- Global groups
- Global groups as members of local groups.

  DB2 for Windows NT enumerates the local and global groups that the user is a member of, using the security database where the user was found. DB2 Universal Database provides an override that forces group enumeration to occur on the local Windows NT server where DB2 is installed, regardless of where the user account was found. This override can be achieved using the following commands:

  – For global settings:

    ```
    db2set -g DB2_GRP_LOOKUP=local
    ```

  – For instance settings:

    ```
    db2set -i DB2_GRP_LOOKUP=local
    ```

  To view all DB2 profile registry variables that are set, type

  ```
  db2set -all
  ```

For DB2 for Windows NT to work with domain security, you must grant authority and privileges to a local group. User names within the local and global groups MUST be defined on the same domain as the local or global group in order to be authenticated correctly.

If the DB2_GRP_LOOKUP profile registry variable is set to local, then DB2 tries to find a user on the local machine only. If the user is not found on the local machine, or is not defined as a member of a local or global group, then

authentication fails. DB2 does **not** try to find the user on another machine in the domain or on the domain controllers.

If the DB2_GRP_LOOKUP profile registry variable is not set then:

1. DB2 first tries to find the user on the same machine.
2. If the user name is defined locally, the user is authenticated locally.
3. If the user is not found locally, DB2 attempts to find the user name on it domain, and then on trusted domains.

The following examples illustrate how DB2 for Windows NT can support domain security. In this first example, the connection works because the user name and local group are on the same domain. In the second example, the connection does not work because the user name and local or global group are on different domains.

Example of a Successful Connection: The connection works in the following scenario because the user name and local or global group are on the same domain.

Note that the user name and local or global group do not need to be defined on the domain where the database server is running, but they must be on the same domain as each other.

Table 24. Successful Connection Using a Domain Controller

Domain1	**Domain2**
A trust relationship exists with Domain2.	• A trust relationship exists with Domain1. • The local or global group `grp2` is defined. • The user name `id2` is defined. • The user name `id2` is part of `grp2`.

The DB2 server runs in this domain. The following DB2 commands are issued from it:
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

The local or global domain is scanned but `id2` is not found. Domain security is scanned.	
	The user name `id2` is found on this domain. DB2 gets additional information about this user name (that is, it is part of the group `grp2`).
The connection works because the user name and local or global group are on the same domain.	

Example of an Unsuccessful Connection: The connection does not work in the following scenario because the user name is defined on a different domain than where the local or global group is defined.

Table 25. Unsuccessful connection using a domain controller

Domain1	**Domain2**
A trust relationship exists with Domain2.	• A trust relationship exists with Domain1. • The local or global group `grp2` is defined.
• The global group `grp1` is defined. • The user name `id1` is defined. • The user name `id1` is part of `grp1`.	
	Domain1\`grp1` is part of `grp2`.

The DB2 server runs in this domain. The following DB2 commands are issued from it:
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

The local or global is scanned and `id1` is found. DB2 gets information for this user name (that is, the user name `id1` is part of `grp1` and the group `grp1` is part of Domain2\`grp2`).	
	The group `grp2` exists on this domain.
The connection does not work because the local or global group is on Domain2 and the actual user name is defined on Domain1. The connection would work if the following command was issue instead: `GRANT CONNECT ON db TO GROUP grp1`	

Appendix F. Using the Windows NT Performance Monitor

There are two performance monitors available to DB2 for Windows NT users:

- **DB2 Performance Monitor**

  The DB2 Performance Monitor provides snapshot and event data related to DB2 and DB2 Connect only. (For more information, click on the **Help** push button in the Control Center and see the Getting Started online help.)
- **Windows NT Performance Monitor**

  The Windows NT Performance Monitor enables you to monitor both database and system performance, retrieving information from any of the performance data providers registered with the system. Windows NT also provides performance information data on all aspects of machine operation including:
  - CPU usage
  - Memory utilization
  - Disk activity
  - Network activity

Registering DB2 with the Windows NT Performance Monitor

The setup program automatically registers DB2 with the Windows NT Performance Monitor for you.

To make DB2 and DB2 Connect performance information accessible to the Windows NT Performance Monitor, you must register the DLL for the DB2 for Windows NT Performance Counters. This also enables any other Windows NT application using the Win32 performance APIs to get performance data.

To install and register the DB2 for Windows NT Performance Counters DLL (DB2Perf.DLL) with the Windows NT Performance Monitor, type:

```
db2perfi -i
```

Registering the DLL also creates a new key in the services option of the registry. One entry gives the name of the DLL, which provides the counter support. Three other entries give names of functions provided within that DLL. These functions include:

- **Open**

  Called when the DLL is first loaded by the system in a process.
- **Collect**

  Called to request performance information from the DLL.

- **Close**

  Called when the DLL is unloaded.

Enabling Remote Access to DB2 Performance Information

If your DB2 for Windows NT workstation is networked to other Windows NT machines, you can use the feature described in this section.

In order to see Windows NT performance objects from another DB2 for Windows NT machine, you must register an administrator username and password with DB2. (The default Windows NT Performance Monitor username, **SYSTEM**, is a DB2 reserved word and cannot be used.) To register the name, type:

```
db2perfr -r username password
```

Note: The `username` used must conform to the DB2 naming rules.

The username and password data is held in a key in the registry, with security that allows access only by administrators and the SYSTEM account. The data is encoded to prevent security concerns about storing an administrator password in the registry.

Notes:

1. Once a username and password combination has been registered with DB2, even local instances of the Performance Monitor will explicitly log on using that username and password. This means that if the username information registered with DB2 does not match, local sessions of the Performance Monitor will not show DB2 performance information.
2. The username and password combination must be maintained to match the username and password values stored in the Windows NT Security database. If the username or password is changed in the Windows NT Security database, the username and password combination used for remote performance monitoring must be reset.
3. To deregister, type:

   ```
   db2perfr -u <username> <password>
   ```

Displaying DB2 and DB2 Connect Performance Values

To display DB2 and DB2 Connect performance values using the Performance Monitor, simply choose the performance counters whose values you want displayed from the **Add to** box. This box displays a list of performance objects providing performance data. Select an object to see a list of the counters it supplies.

A performance object can also have multiple instances. For example, the LogicalDisk object provides counters such as "% Disk Read Time" and "Disk Bytes/sec"; it also has an instance for each logical drive in the machine, including "C:" and "D:".

Windows NT provides the following performance objects:

- **DB2 Database Manager**

  This object provides general information for a single Windows NT instance. The DB2 instance being monitored appears as the object instance.

  For practical and performance reasons, you can only get performance information from one DB2 instance at a time. The DB2 instance that the Performance Monitor shows is governed by the `db2instance` registry variable in the Performance Monitor process. If you have multiple DB2 instances running simultaneously and want to see performance information from more than one, you must start a separate session of the Performance Monitor, with `db2instance` set to the relevant value for each DB2 instance to be monitored.

  If you are running a partitioned database system, you can only get performance information from one database partition server (node) at a time. By default, the performance information for the default node (i.e. the node that has logical port 0) is displayed. To see performance information of another node, you must start a separate session of the Performance Monitor with the DB2NODE environment variable set to the node number of the node to be monitored.

- **DB2 Databases**

  This object provides information for a particular database. Information is available for each currently active database.

- **DB2 Applications**

  This object provides information for a particular DB2 application. Information is available for each currently active DB2 application.

- **DB2 DCS Databases**

  This object provides information for a particular DCS database. Information is available for each currently active database.

- **DB2 DCS Applications**

  This object provides information for a particular DB2 DCS application. Information is available for each currently active DB2 DCS application.

Which of these objects will be listed by the Windows NT Performance Monitor depends on what is installed on your Windows NT machine and what applications are active. For example, if DB2 UDB is installed and the database manager has been started, the DB2 Database Manager object will be listed. If there are also some DB2 databases and applications currently active on that machine, the DB2 Databases and DB2 Applications objects will be

listed as well. If you are using your Windows NT system as a DB2 Connect gateway and there are some DCS databases and applications currently active, the DB2 DCS Databases and DB2 DCS Applications objects will be listed.

Accessing Remote DB2 Performance Information

Enabling remote access to DB2 Performance Information was discussed earlier. In the **Add to** box, select another computer to monitor. This brings up a list of all the available performance objects on that computer.

In order to be able to monitor DB2 Performance object on a remote computer, the level of the DB2 UDB or DB2 Connect code installed on that computer must be Version 6 or higher.

Resetting DB2 Performance Values

When an application calls the DB2 monitor APIs, the information returned is normally the cumulative values since the DB2 server was started. However, often it is useful to:

- Reset performance values
- Run a test
- Reset the values again
- Re-run the test.

To reset database performance values, use the **db2perfc** program. Type:

```
db2perfc
```

By default, this resets performance values for all active DB2 databases. However, you can also specify a list of databases to reset. You can also use the -d option to specify that performance values for DCS databases should be reset. For example:

```
db2perfc
db2perfc dbalias1 dbalias2 ... dbaliasn

db2perfc  -d
db2perfc  -d  dbalias1 dbalias2 ... dbaliasn
```

The first example resets performance values for all active DB2 databases. The next example resets values for specific DB2 databases. The third example resets performance values for all active DB2 DCS databases. The last example resets values for specific DB2 DCS databases.

The **db2perfc** program resets the values for ALL programs currently accessing database performance information for the relevant DB2 server instance (that is, the one held in `db2instance` in the session in which you run **db2perfc**.

Invoking **db2perfc** also resets the values seen by anyone remotely accessing DB2 performance information when the **db2perfc** command is executed.

Note: There is a DB2 API, `sqlmrset`, that allows an application to reset the values it sees locally, not globally, for particular databases. See *Administrative API Reference* for more information.

Appendix G. Working with Windows NT or Windows 2000 Database Partition Servers

When working to change the characteristics of your configuration in a Windows NT or Windows 2000 environment, the tasks involved are carried out using specific utilities. Other operating system environments use the methods shown in the "Scaling Your Configuration Through Adding Processors" chapter of the *Administration Guide: Performance*.

The utilities presented here are:

- "Listing Database Partition Servers in an Instance"
- "Adding a Database Partition Server to an Instance"
- "Changing the Database Partition" on page 377
- "Dropping a Database Partition From an Instance" on page 378

Listing Database Partition Servers in an Instance

On Windows NT or Windows 2000, use the **db2nlist** command to obtain a list of database partition servers that participate in an instance.

The command is used as follows:

```
db2nlist
```

When using this command as shown, the default instance is the current instance (set by the DB2INSTANCE environment variable). To specify a particular instance, you can specify the instance using:

```
db2nlist /i:instName
```

where `instName` is the particular instance name you want.

You can also optionally request the status of each partition server by using:

```
db2nlist /s
```

The status of each database partition server may be one of: starting, running, stopping, or stopped.

Adding a Database Partition Server to an Instance

On Windows NT or Windows 2000, use the **db2ncrt** command to add a database partition server (node) to an instance.

Note: Do not use the **db2ncrt** command if the instance already contains databases. Instead, use the **db2start addnode** command. This ensures that the database is correctly added to the new database partition server. **DO NOT EDIT** the `db2nodes.cfg` file, since changing the file may cause inconsistencies in the partitioned database system.

The command has the following required parameters:

```
db2ncrt /n:node_number
        /u:username,password
        /p:logical_port
```

- /n:

  The unique node number to identify the database partition server. The number can be from 1 to 999 in ascending sequence.

- /u:

  The logon account name and password of the DB2 service.

- /p:logical_port

  The logical port number used for the database partition server if the logical port is not zero (0). If not specified, the logical port number assigned is 0.

The logical port parameter is only optional when you create the first node on a machine. If you create a logical node, you must specify this parameter and select a logical port number that is not in use. There are several restrictions:

- On every machine there must be a database partition server with a logical port 0.
- The port number cannot exceed the port range reserved for FCM communications in the services file in `x:\winnt\system32\drivers\etc\` directory. For example, if you reserve a range of four ports for the current instance, then the maximum port number would be 3 (ports 1, 2, and 3; port 0 is for the default logical node). The port range is defined when **db2icrt** is used with the `/r:base_port, end_port` parameter.

There are also several optional parameters:

- /g:network_name

  Specifies the network name for the database partition server. If you do not specify this parameter, DB2 uses the first IP address it detects on your system.

  Use this parameter if you have multiple IP addresses on a machine and you want to specify a specific IP address for the database partition server. You can enter the *network_name* parameter using the network name or IP address.

- /h:host_name

The TCP/IP host name that is used by FCM for internal communications if the host name is not the local host name. This parameter is required if you add the database partition server on a remote machine.

- /i:instance_name

  The instance name; the default is the current instance.

- /m:machine_name

  The computer name of the Windows NT workstation on which the node resides; the default name is the computer name of the local machine.

- /o:instance_owning_machine

  The computer name of the machine that is the instance-owning machine; the default is the local machine. This parameter is required when the **db2ncrt** command is invoked on any machine that is not the instance-owning machine.

For example, if you want to add a new database partition server to the instance TESTMPP (so that you are running multiple logical nodes) on the instance-owning machine MYMACHIN, and you want this new node to be known as node 2 using logical port 1, enter:

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

Changing the Database Partition

On Windows NT or Windows 2000, use the **db2nchg** command to do the following:

- Move the database partition from one machine to another.
- Change the TCP/IP host name of the machine.

  If you are planning to use multiple network adapters, you must use this command to specify the TCP/IP address for the "netname" field in the *db2nodes.cfg* file.
- Use a different logical port number.
- Use a different name for the database partition server (node).

The command has the following required parameter:

```
db2nchg /n:node_number
```

The parameter /n: is the node number of the database partition server's configuration you want to change. This parameter is required.

Optional parameters include:

- /i:instance_name

  Specifies the instance that this database partition server participates in. If you do not specify this parameter, the default is the current instance.

- /u:username,password

  Changes the logon account name and password for the DB2 service. If you do not specify this parameter, the logon account and password remain the same.
- /p:logical_port

  Changes the logical port for the database partition server. This parameter must be specified if you move the database partition server to a different machine. If you do not specify this parameter, the logical port number remains unchanged.
- /h:host_name

  Changes the TCP/IP hostname used by FCM for internal communications. If you do not specify this parameter, the hostname is unchanged.
- /m:machine_name

  Moves the database partition server to another machine. The database partition server can only be moved if there are no existing databases in the instance.
- /g:network_name

  Changes the network name for the database partition server.

  Use this parameter if you have multiple IP addresses on a machine and you want to use a specific IP address for the database partition server. You can enter the `network_name` using the network name or the IP address.

For example, to change the logical port assigned to node 2, which participates in the instance TESTMPP, to use the logical port 3, enter the following command:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

Dropping a Database Partition From an Instance

On Windows NT or Windows 2000, use the **db2ndrop** command to drop a database partition server (node) from an instance that has no databases. If you drop a database partition server, its node number can be reused for a new database partition server.

Exercise caution when you drop database partition servers from an instance. If you drop the instance-owning database partition server node zero (0) from the instance, the instance will become unusable. If you want to drop the instance, use the **db2idrop** command.

Note: Do not use the **db2ndrop** command if the instance contains databases. Instead, use the **db2stop drop nodenum** command. This ensures that the database is correctly removed from the database partition. **DO**

NOT EDIT the db2nodes.cfg file, since changing the file may cause inconsistencies in the partitioned database system.

If you want to drop a node that is assigned the logical port 0 from a machine that is running multiple logical nodes, you must drop all the other nodes assigned to the other logical ports before you can drop the node assigned to logical port 0. Each database partition server must have a node assigned to logical port 0.

The command has the following parameters:

```
db2ndrop /n:node_number /i:instance_name
```

- /n:

  The unique node number to identify the database partition server. This is a required parameter. The number can be from zero (0) to 999 in ascending sequence. Recall that node zero (0) represents the instance-owning machine.

- /i:instance_name

  The instance name. This is an optional parameter. If not given, the default is the current instance (set by the DB2INSTANCE registry variable).

Appendix H. Configuring Multiple Logical Nodes

Typically, you configure DB2 Enterprise - Extended Edition to have one database partition server assigned to each machine. There are several situations, however, in which it would be advantageous to have several database partition servers running on the same machine. This means that the configuration can contain more nodes than machines. In these cases, the machine is said to be running *multiple logical nodes* if they participate in the *same* instance. If they participate in different instances, this machine is *not* hosting multiple logical nodes.

With multiple logical node support, you can choose from three types of configurations:

- A standard configuration, where each machine has only one database partition server.
- A multiple logical node configuration, where a machine has more than one database partition server.
- A configuration where several logical nodes run on each of several machines.

Configurations that use multiple logical nodes are useful when the system runs queries on a machine that has symmetric multiprocessor (SMP) architecture. The ability to configure multiple logical nodes on a machine is also useful if a machine fails. If a machine fails (causing the database partition server or servers on it to fail), you can restart the database partition server (or servers) on another machine using the DB2START NODENUM command. This ensures that user data remains available.

Another benefit is that multiple logical nodes can exploit SMP hardware configurations. In addition, because database partitions are smaller, you can obtain better performance when performing such tasks as backing up and restoring database partitions and table spaces, and creating indexes.

You can configure multiple logical nodes in one of two ways:

- Configure the logical nodes (database partitions) in the `db2nodes.cfg` file. You can then start all the logical and remote nodes with the DB2START command or its associated API.

  Note: For Windows NT, you must use *db2ncrt* to add a node if there is no database in the system; or, DB2START ADDNODE command if there is one or more databases. Within Windows NT, the *db2nodes.cfg* file should never be manually edited.

- Restart a logical node on another processor on which other logical database partitions (nodes) are already running. This allows you to override the hostname and port number specified for the logical database partition in `db2nodes.cfg`.

To configure a logical database partition (node) in `db2nodes.cfg`, you must make an entry in the file to allocate a logical port number for the node. Following is the syntax you should use:

```
nodenumber hostname logical-port netname
```

Note: For Windows NT, you must use *db2ncrt* to add a node if there is no database in the system; or, DB2START ADDNODE command if there is one or more databases. Within Windows NT, the *db2nodes.cfg* file should never be manually edited.

The format for the *db2nodes.cfg* file on Windows NT is different when compared to the same file on Unix. On Windows NT, the column format is:

```
nodenumber hostname computername logical_port netname
```

You must ensure that you define enough ports in the `services` file of the `etc` directory for FCM communications.

Appendix I. High Speed Inter-Node Communications

When using DB2 Universal Database Enterprise - Extended Edition, you may be working in a communication-intensive environment where overall system throughput is vital to your business.

There are two types of networks that may be used for your partitioned environment. One uses TCP/IP over a public LAN. The other type uses TCP/IP or the Virtual Interface (VI) architecture over a dedicated interconnect.

The public interconnect works with existing TCP/IP. TCP/IP is available as a communication protocol almost everywhere. This is a Local Area Network (LAN) environment. An advantage with this environment is that you can choose to immediately attach your cluster without requiring additional proprietary hardware and software. A disadvantage with this environment is that the additional cluster traffic affects the quality of the service over the entire LAN. For example, there could be a communication "burst" effect with database activity within the cluster that affects communication across the LAN. Also, the communications of the rest of the LAN environment makes it difficult to maintain consistent performance of database processing within the cluster.

The dedicated interconnect works as a separate network. The network may be the only network available for use within the cluster or it may be used in addition to the LAN environment. The network is dedicated to providing communications between the members of the cluster. This is called a System Area Network (SAN). Performance of the database is not affected by external communications traffic (as in a LAN environment), and the reverse is also true. A disadvantage with this environment is that there may be separate administration required for both networks as well as additional separate hardware, software, and protocol costs for both the LAN and the SAN. An example of a dedicated interconnect is the 100 Mb/Sec Ethernet.

You may wish to maintain your pre-existing public LAN environment, but you also want the ability to bulk transfer data over the SAN (within your cluster). Such an arrangement is convenient if you want to have communication access beyond the cluster. Within a Windows NT operating environment, you may need to retain the public LAN for communication access to the NT Domain Controller. (See "Appendix E. How DB2 for Windows NT Works with Windows NT Security" on page 361 for information on the Domain Controller.)

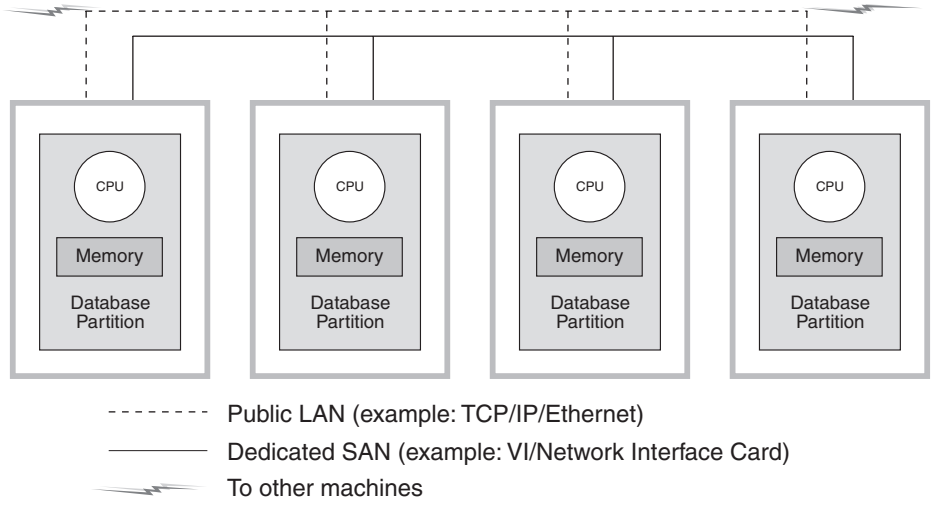


Figure 12. Combine Dedicated SAN with Public LAN

The remaining sections include discussions of:

- "High Speed Interconnection Using TCP/IP"
- "High Speed Interconnection Using VI" on page 385

High Speed Interconnection Using TCP/IP

Examples of the prerequisites for the network hardware setup using TCP/IP are:

- Standard Ethernet.

  There are no unusual hardware, software, or protocol requirements.
- IBM Netfinity SP Switch.

  The requirements are outlined in the next section.

Prerequisites for Using an IBM Netfinity SP Switch

To find out about Netfinity, please use the following URL:
http://www.ibm.com/pc/us/netfinity

For additional documentation and software upgrades, go to the IBM Support Web site at: http://www.ibm.com/pc/support

1. Click **Servers**
2. Under family, click **Clustering**
3. Under Technical Information, click **Downloadable files** for software upgrades or click **Online publications** for documentation

Locate the IBM Netfinity SP Switch topic and download the files needed.

Setup Procedure for an IBM Netfinity SP Switch

Directions for installing the IBM Netfinity SP Switch are found in the *IBM Netfinity SP Switch Installation and User's Guide*.

You should use the hardware and software guides that come with the various hardware and software components (like the server rack enclosure, the host adapter, and the SP switch software) to install, configure, and test those components.

DB2, once installed and without any additional modifications, will use the IBM Netfinity SP Switch.

High Speed Interconnection Using VI

Virtual Interface (VI) Architecture is the inter-node communication protocol alternative to TCP/IP in a Windows NT massively parallel processing (MPP) environment. VI is a new communication architecture that was developed jointly by Intel, Microsoft, and Compaq to improve performance over a System Area Network (SAN). Refer to http://www.viarch.org for more information on the architecture.

Products exist which may be acquired separately from DB2 Universal Database that have a VIA-enabled network interface card (NIC), switch, and software driver implementation. Several Independent Hardware Vendors (IHVs) have released, or plan to release, such products.

VI Architecture has low latency, high bandwidth, and lower CPU consumption when compared to TCP/IP. In a communication-intensive environment, using VI Architecture improves the overall system throughput. The greater the number of nodes in the cluster, and the greater the amount of data transferred, the greater the benefit from using VI Architecture.

DB2 Universal Database supports VI Architecture implementations that comply with the *Virtual Interface Architecture Specification, Version 1.0*, the *Intel Virtual Interface (VI) Architecture Developers' Guide, Version 1.0*, and pass the "Virtual Interface Architecture Conformance Suite". The specification is found at http://www.intel.com/design/servers/vi/the_spec/specification.htm on the Web. The Developer's Guide is found at http://www.intel.com/design/servers/vi/developer/ia_imp_guide.htm on the Web. Information on the conformance suite is also found at this same URL.

IBM announced support for Virtual Interface (VI) Architecture with DB2 Universal Database EEE V5.2.

To find out about other products adhering to VI Architecture and supported by DB2 Universal Database EEE, please contact the DB2 Universal Database support organization at http://www.software.ibm.com/data or call 1-800-237-5511 (only in the U.S.A. and Canada).

The products that have been tested with DB2 Universal Database include:

- GigaNet Interconnect, see "Setup Procedure for GigaNet Interconnect" for details.
- Compaq ServerNet Interconnect, see "Setup Procedure for ServerNet Interconnect" on page 389 for details.
- Fujitsu Synfinity Interconnect, see "Setup Procedure for Synfinity Interconnect" on page 392 for details.

There may be other products that work with DB2 Universal Database. Check with the vendor of that product, and with IBM Service and Support, to ensure that the other product is supported.

Virtual Interface (VI) Hardware Setup

Examples of the prerequisites for the network hardware setup using VI are:

- GigaNet Interconnect.

  "Setup Procedure for GigaNet Interconnect" has the overview of the hardware, software, and protocol information needed for this choice.

  To find out about GigaNet products, or to contact GigaNet Service and Support, please use the following URL: http://www.giganet.com/
- Compaq ServerNet Interconnect.

  "Setup Procedure for ServerNet Interconnect" on page 389 has the overview of the hardware, software, and protocol information needed for this choice.

  To find out about ServerNet products, or to contact ServerNet Service and Support, please use the following URL: http://www.servernet.com/
- Fujitsu Synfinity Interconnect.

  "Setup Procedure for Synfinity Interconnect" on page 392 has the overview of the hardware, software, and protocol information needed for this choice.

  To find out about Synfinity products, or to contact Synfinity Service and Support, at Fujitsu System Technologies, please use the following URL: http://www.fujitsu.com/

You must configure DB2 to use VI. "Enabling DB2 to Run Using VI" on page 393 has the necessary information for you to use VI.

Setup Procedure for GigaNet Interconnect

The list of the hardware and software required to set up this environment include the following products:

- GigaNet GNN1000 Network Interface Card

- GigaNet GNX5000 Switch
- GigaNet GNCxx11 Copper Interconnect Cables
- GigaNet cLAN Software, Version 2.0.

The steps required to ensure that GigaNet Interconnect can work with DB2 Universal Database are shown below. Each step is a summary of what is required at each step: all of the details associated with each step are not presented here. You should also use the referenced documentation at each step which does provide detailed instructions and direction needed.

Each GigaNet GNN1000 is packaged with a GigaNet cLAN Software CD-ROM. The CD-ROM contains all of the necessary software to set up the GigaNet Interconnect. In addition, the CD-ROM also contains the VI Architecture software developer's kit (SDK) and the Adobe Acrobat Reader. The VI Architecture SDK will be used to develop and test VI-enabled applications. The Adobe Acrobat Reader will be used to view documents on the CD-ROM that explain how to develop VI-enabled applications.

Summary of steps:

1. Install Adapter Cards
2. Install Switches and Cables
3. Install Adapter Drivers
4. Install cLAN Management Console
5. Test the Interconnect

Here are the steps:

1. Install the GigaNet GNN1000 Network Interface Card. Please refer to the *GigaNet GNN1000 User Guide* for installation instructions.
2. Install the GigaNet GNX5000 Switch and Cables. Please refer to the *GigaNet GNX5000 User Guide* for installation instructions.
3. Install the GigaNet GNN1000 Adapter Driver software on each node connected to the GNX5000 Switch. Please refer to the *GigaNet GNN1000 User Guide* for installation instructions. Here are additional details if you are installing drivers provided by GigaNet:
   a. Remove any previous version of the GNN1000 Driver already installed. Removal requires the node to be re-booted.
   b. Use `Start–>Setting–>Control Panel–>Networks–>Adapters–>Add` to install the driver.
   c. Click **Have Disk...** and specify the Driver directory on the CD-ROM. For example, if F: is your CD-ROM drive, then you would use `F:\Driver`
   d. Select "GNN1000 NDIS Adapter" and then click **OK**.

e. Configure Network protocols to complete the installation.

GigaNet Adapter Driver software is also available on GigaNet's web site, http://www.giganet.com. Please refer to the download and installation instructions found on the support page of GigaNet's web site.

The installation of the GNN1000 Adapter Driver causes the node to re-boot.

4. The GigaNet cLAN Management Console (GMC) can be used to test the integrity of the GigaNet Interconnect. The GigaNet cLAN Management Console is comprised of two parts: the Console, and the Agent. The Agent must be installed on all nodes in the cluster. The Console can be installed on any network node that has access to the nodes in the cluster. The most versatile and recommended installation is that which has both the Console and the Agent installed on each node in the cluster.

   Install the GigaNet cLAN Management Console. Please refer to the *GigaNet GNN1000 User Guide* for installation instructions and additional information about the cLAN Management Console. Here are additional details on the installation procedure:

   a. Insert the cLAN Software CD into the CD-ROM drive.
   b. Wait for the CD automatic installation menu to appear.
   c. Click on "Install cLAN Management Console."
   d. Repeat this installation procedure on each remaining node in the cluster.

   GigaNet cLAN Management Console software is also available on GigaNet's web site, http://www.giganet.com. Please refer to the download and installation instructions found on the support page of GigaNet's web site.

   The installation of the cLAN Management Console may cause the node to re-boot.

5. Test that the GigaNet Hardware is working. This can be done by doing the following:

   a. Open the GMC. (`Programs->GigaNet->cLAN Management Console`)
   b. A dialog box is displayed showing all accessible machines in the LAN. Press **ESC**.
   c. Select **Console—>Local** from the menu bar.
   d. Confirm that all the members in the cluster are shown and that they are all "Active".
   e. Select **Utilities—>VI Throughput** from the menu bar. This will run a throughput test to check that the data is actually going through the hardware.

f. Enter in uppercase letters the computer names of the two nodes you wish to use in the test. Identify the local node as the source node.

g. Click **Start Measuring**. You should see data being transferred at a rate of at least 65 MB per second.

h. Click **Stop Measuring** to stop the connection test.

i. Repeat the test for the other nodes in the cluster by measuring throughput between the local node (Source) and the other nodes (Sink).

If the connection test does not appear to be working, refer to the troubleshooting sections of the *GigaNet GNN1000 User Guide* and the *GigaNet GNX5000 User Guide*.

Refer to *DB2 Enterprise - Extended Edition for Windows Quick Beginnings* for information on how to install and implement DB2 Universal Database for Windows NT.

Setup Procedure for ServerNet Interconnect

The list of the hardware and software required to set up this environment include the following products:

- ServerNet PCI Adapter Driver (SPAD), (product ID T0089), version 1.3.5 or later
- ServerNet Switch 1
- ServerNet Area Network Manager (SANMan), (product ID T0087), version 1.1.3 or later.

The following are the steps required to ensure that ServerNet Interconnect can work with DB2 Universal Database. Each step is a summary of what is required at each step: all of the details associated with each step are not presented here. You should also use the referenced documentation at each step which does provide detailed instructions and direction needed.

The steps shown below also assume that you are only using up to six (6) nodes in the cluster. Contact ServerNet if you have a requirement to use more than six nodes.

Here are the steps:

1. Install the ServerNet Network Interface Card. Please refer to the *ServerNet-I Virtual Interface Software Release Document, (product ID N0031)* for installation instructions.
2. Install the ServerNet Switch 1. Please refer to the *ServerNet-I Virtual Interface Software Release Document, (product ID N0031)* for installation instructions.
3. Uninstall previous ServerNet drivers. (Skip this step if this is your first time installing ServerNet.)

a. Open the Network control panel. (`Start–>Setting–>Control Panel–>Network`)
b. Click on the **Adapters Tab**.
c. Remove Tandem ServerNet PCI Adapter Driver.
d. Click on the **Services Tab**.
e. Remove SANMan.
f. Click on the **Protocols Tab**.
g. Remove Tandem ServerNet-I VI Protocol.

4. Install the Tandem ServerNet PCI Adapter Driver. Here are additional details if you are installing using the software CD provided by ServerNet:
   a. Open the Network control panel. (`Start–>Setting–>Control Panel–>Network`)
   b. Click on the **Adapters Tab**. (The Adapters screen appears.)
   c. Ensure the new ServerNet driver is placed in a separate drive and/or directory. Then, from the command prompt referencing the correct drive and/or directory, type "ernnn.exe -d" to start the self-extracting program. ("ernnn.exe" is the name of the Engineering Release followed by a number — ERnnn.EXE — that identifies the specific version of the ServerNet driver to be installed.)
   d. Change to the drive and/or directory where the extracted files are located. Change to the "Spad n.n.n \ Free" subdirectory (where "n.n.n" is the specific version of the product). (If you are working in a troubleshooting or a development environment, then change to the "Spad n.n.n \ Checked" subdirectory instead of the "Spad n.n.n \ Free" subdirectory.)
   e. Rename the "oemsetup.multi_node" file to "oemsetup.inf".
   f. Choose **Add** in the Adapters Tab. (The Select Adapters screen appears.)
   g. Click **Have Disk...**. (The Insert Disk screen appears.)
   h. Enter the drive and/or directory where the oemsetup.inf file is located.
   i. Ensure the dialog box shows "Tandem ServerNet PCI Adapter Driver" and then click **OK**. Ensure the list of adapters now shows the ServerNet adapter. Click **Close**.
   j. Choose **Yes** to restart the computer. Or, select **No** and continue installing SANMan and the VI Software Developer's Kit (SDK).
5. Install SANMan. Here are additional details if you are installing using the software CD provided by ServerNet:
   a. Open the Network control panel. (`Start–>Setting–>Control Panel–>Network`)
   b. Click on the **Services Tab**. (The Services screen appears.)
   c. Ensure the new ServerNet driver is placed in a separate drive and/or directory. Then, from the command prompt referencing the correct

drive and/or directory, type “ernnn.exe -d” to start the self-extracting program. (“ernnn.exe” is the name of the Engineering Release followed by a number — ERnnn.EXE — that identifies the specific version of the ServerNet driver to be installed.)

d. Choose **Add** in the Services Tab. (The Select Services screen appears.)

e. Change to the drive and/or directory where the extracted files are located. Change to the “SANMan n.n.n \Free” subdirectory (where “n.n.n” is the specific version of the product). (If you are working in a troubleshooting or a development environment, then change to the “SANMan n.n.n \ Checked” subdirectory instead of the “SANMan n.n.n \ Free” subdirectory.)

f. Determine if the Switch is X or Y by looking at the light on the Switch. One light says “X”, and the one light says “Y”.

g. If an X Switch, select X=1 and Y=0. Ensure all cables are connected to the X port on the network cards.

h. If a Y Switch, select X=0 and Y=1. Ensure all cables are connected to the Y port on the network cards.

i. Provide the port number of the switch to which the network card on the current machine is connected.

j. Select “PC” for all six (6) ports.

6. Install the Virtual Interface Protocol. Here are additional details if you are installing using the software CD provided by ServerNet:

   a. Open the Network control panel. (`Start–>Setting–>Control Panel–>Network`)

   b. Click on the **Protocols Tab**. (The Network Protocols screen appears.)

   c. Ensure the new ServerNet driver is placed in a separate drive and/or directory. Then, from the command prompt referencing the correct drive and/or directory, type “ernnn.exe -d” to start the self-extracting program. (“ernnn.exe” is the name of the Engineering Release followed by a number — ERnnn.EXE — that identifies the specific version of the ServerNet driver to be installed.)

   d. Choose **Add** in the Protocols Tab. (The Select Network Protocols screen appears.)

   e. Click **Have Disk...**. (The Insert Disk screen appears.)

   f. Enter the drive and/or directory where the extracted files are located.

7. Test that the ServerNet Hardware is working. There are no test programs available. Instead, simply use DB2 to test the ServerNet hardware.

   If the hardware does not appear to be working, refer to the *ServerNet-I Virtual Interface Software Release Document, (product ID N0031)* for additional troubleshooting help.

Refer to *DB2 Enterprise - Extended Edition for Windows Quick Beginnings* for information on how to install and implement DB2 Universal Database for Windows NT.

Setup Procedure for Synfinity Interconnect

The list of the hardware and software required to set up this environment include the following products:

- Synfinity PCI Network Interface Card
- Synfinity Six Port Switch
- Synfinity Interconnect Cables
- Synfinity Cluster Manager Software, Version 1.10.

The steps required to ensure that Synfinity Interconnect can work with DB2 Universal Database are shown below. Each step is a summary of what is required at each step: all of the details associated with each step are not presented here. You should also use the referenced documentation at each step which does provide detailed instructions and the direction needed.

Each Synfinity System is packaged with a Synfinity Cluster Manager Software, Version 1.10 CD-ROM. The CD-ROM contains all of the necessary documentation and software to set up the Synfinity Interconnect. In addition, the CD-ROM also contains the *Synfinity Cluster Manager Software User Guide*.

If you have other VI hardware, software, and protocol installed, it may be necessary to remove all of them before installing your Synfinity interconnect.

Once installed, Synfinity interconnect is considered to be exotic hardware and may not be viewed through the Windows NT control panel.

Summary of steps:

1. Install Adapter Cards
2. Install Synfinity Cluster Manager Software
3. Install Switches and Cables
4. Test the Interconnect

Here are the steps:

1. Install the Synfinity PCI Network Interface Card. Please refer to the *Synfinity Cluster Manager Software User Guide* for installation instructions.
2. Install the Synfinity Cluster Manager Software on a node connected to the Switch. Please refer to the *Synfinity Cluster User Guide* for installation instructions.

   The node you select will be the Cluster Manager. This is the only node where you have to install the software from the CD.

Once installed, you should run the Synfinity Cluster Manager software. The Cluster Manager will give you a cluster plan and help you through a step-by-step guide to configuring the network, and advise the best routing and cabling options. This step should be completed before any cables are connected to the Synfinity switches and network cards. As part of the planning process, the Cluster Manager will use the cluster plan to create installable diskettes for use on the other nodes. This will include the driver software for the cards that are on the other nodes. Refer to the *Synfinity Cluster Manager Software User Guide* for complete details.

3. Install the Synfinity Switch and Cables. Please refer to the *Synfinity Cluster User Guide* for installation instructions.
4. Test that the Synfinity Hardware is working. This can be done by doing the following:
   a. On any system in the cluster, open a ″Command Prompt″ window in Windows NT.
   b. Change directory to the ″utils″ subdirectory of where the Synfinity Cluster Manager software was loaded.
   c. Type ″vitest″ and note the node number that is displayed.
   d. Move to any other system in the cluster, open a ″Command Prompt″ window.
   e. Change directory to the ″utils″ subdirectory of where the Synfinity Cluster Manager software was loaded on this other system.
   f. Type ″vitest x″ where x is the node number from step c above.
   g. A ″CONNECTION GOOD″ message should be displayed.
   h. If a ″NO CONNECTION″ message is displayed, check cabling and hardware set up, and refer to the *Synfinity Cluster Manager Software User Guide* for further information troubleshooting the problem. Also check the support web pages for ″Tech-tips″ at http://www.fujitsu.com/

Refer to *DB2 Enterprise - Extended Edition for Windows Quick Beginnings* for information on how to install and implement DB2 Universal Database for Windows NT.

Enabling DB2 to Run Using VI

Detailed installation information is found in *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

After completing the installation of DB2 as documented in *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*, set the following DB2 registry variables and carry out the following tasks on each database partition in the instance:

1. Set DB2_VI_ENABLE=ON

Use the **db2set** command to modify the value for the registry variable. Use the **db2_all** command to run the **db2set** command on all database partition servers in the instance. You must be logged on with a user account that is a member of the Administrators group to run the **db2_all** command.

In the following example, the ; character is placed inside the double quotation marks to allow the request to run concurrently on all the database partition servers in the instance:

```
db2_all ";db2set DB2_VI_ENABLE=ON"
```

For more information about the **db2_all** command, see "Issuing Commands to Multiple Database Partition Servers" in the *Administration Guide: Implementation*.

2. Set DB2_VI_DEVICE=nic0

   For example:

   ```
   db2_all ";db2set DB2_VI_DEVICE=nic0"
   ```

   Note: With Synfinity Interconnect, this variable should be set `DB2_VI_DEVICE=VINIC`. The device name (VINIC) must be in upper case.

3. Set DB2_VI_VIPL=vipl.dll

   For example:

   ```
   db2_all ";db2set DB2_VI_VIPL=vipl.dll"
   ```

   Note: The value used in the example is the default for the registry variable. For more information on the registry variables, see *Administration Guide: Performance*.

4. Enter `db2start` on the partitioned instance.
5. Review the db2diag.log file. There should be one message for each partition stating that "VI is enabled."
6. Fast Communications Manager (FCM) configuration parameters may need to be updated. Should you encounter a problem as a result of resource constraints involving FCM, you should raise the values of the FCM configuration parameters. If you are moving from another high speed interconnect environment where you have increased the values for the FCM configuration parameters, you may need to lower these values. Also, on Windows NT, you may be required to set the DB2NTMEMSIZE registry variable to override the DB2 defaults. Refer to *Administration Guide: Performance* for more information on the registry variables.

Appendix J. Lightweight Directory Access Protocol (LDAP) Directory Services

Lightweight Directory Access Protocol (LDAP) is an industry standard access method to directory services. A directory service is a repository of resource information about multiple systems and services within a distributed environment; and it provides client and server access to these resources. Each database server instance will publish its existence to an LDAP server and provide database information to the LDAP directory when the databases are created. When a client connects to a database, the catalog information for the server can be retrieved from the LDAP directory. Each client is no longer required to store catalog information locally on each machine. Client applications search the LDAP directory for information required to connect to the database.

A caching mechanism exists so that the client only searches the LDAP directory once in its local directory catalogs. Once the information is retrieved, it is stored or cached on the local machine. Subsequent access to the same information is based on the values of the *dir_cache* database manager configuration parameter and the DB2LDAPCACHE registry variable.

- If DB2LDAPCACHE=NO and *dir_cache*=NO, then always read the information from LDAP.
- If DB2LDAPCACHE=NO and *dir_cache*=YES, then read the information from LDAP once and insert it into the DB2 cache.
- If DB2LDAPCACHE=YES or is not set, and if the required information is not found in the local cache, then the information is read from the LDAP directory and the local cache is refreshed.

Note: The DB2LDAPCACHE registry variable is only applicable to the database and node directories.

Supporting LDAP Client and Server Configurations

The following table summarizes the supported LDAP client and server configurations:

Table 26. Supported LDAP Client and Server Configurations

	IBM SecureWay Directory V3.1 and V3.1.1	**Microsoft Active Directory**
IBM LDAP Client	Supported	Not supported
Microsoft LDAP/ADSI Client	Supported	Supported

IBM SecureWay Directory Version 3.1 is an LDAP Version 3 server available for Windows NT, AIX, and Solaris. SecureWay directory is shipped as part of the base operating system on AIX and AS/400, and with OS/390 Security Server.

DB2 supports IBM LDAP client on AIX, Solaris, Windows NT, and Windows 98.

Microsoft Active Directory is an LDAP Version 3 server and is available as part of the Windows 2000 Server operating system.

The Microsoft LDAP Client support is included in the following Microsoft products:

- Outlook 98, Outlook 2000, or Outlook Express

  Note: Outlook Express is installed as part of Microsoft Internet Explorer.
- Exchange Server
- Windows NT Server Service Pack 4
- Windows 98 Second Edition
- Windows 2000

The Microsoft LDAP Client support is also included in the Active Directory Service Interface (ADSI) component. The latest version of ADSI can be downloaded from

http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/adsilinks.asp

When running on Windows 98, Windows NT, or Windows 2000 operating systems, DB2 supports using either the IBM LDAP client or the Microsoft LDAP client to access the IBM SecureWay Directory Server. If the Microsoft LDAP client is not available, DB2 attempts to use the IBM LDAP client. To explicitly select the IBM LDAP client, use the **db2set** command to set the DB2LDAP_CLIENT_PROVIDER registry variable to "IBM".

Support for Windows 2000 Active Directory

DB2 exploits the Active Directory as follows:

1. The DB2 database servers are published in the Active Directory as the ibm_db2Node objects. The ibm_db2Node object class is a subclass of the ServiceConnectionPoint (SCP) object class. Each ibm_db2Node object contains protocol configuration information to allow client applications to connect to the DB2 database server. When a new database is created, the

database is published in the Active Directory as the ibm_db2Database object under the ibm_db2Node object.

2. When connecting to a remote database, DB2 client queries the Active Directory, via the LDAP interface, for the ibm_db2Database object. The protocol communication to connect to the database server (binding information) is obtained from the ibm_db2Node object which the ibm_db2Database object is created under.

Configuring DB2 to Use Active Directory

In order to access Microsoft Active Directory, ensure that the following conditions are met:

1. The machine that runs DB2 must belong to a Windows 2000 domain.
2. The Microsoft LDAP client is installed. Microsoft LDAP client is part of the Windows 2000 operating system. For Windows 98, or Windows NT, you need to verify that the `wldap32.dll` exists under the system directory.
3. Enable the LDAP support. For Windows 2000, the LDAP support is enabled by the installation program. For Windows 98/NT, you must explicitly enable LDAP by setting the DB2_ENABLE_LDAP registry variable to "YES" using the **db2set** command.
4. Log on to a domain user account when running DB2 to read information from the Active Directory.

Configuring DB2 in the IBM LDAP Environment

Before you can use DB2 in the IBM LDAP environment, you must configure the following on each machine:

- Enable the LDAP support. For Windows 2000, the LDAP support is enabled by the installation program. For Windows 98/NT, you must explicitly enable LDAP by setting the DB2_ENABLE_LDAP registry variable to "YES" using the **db2set** command.
- The LDAP server's TCP/IP host name and port number. These values can be entered during unattended installation using the DB2LDAPHOST response keyword, or you can manually set them later by using the DB2SET command:

  ```
  db2set DB2LDAPHOST=<hostname[:port]>
  ```

  where `hostname` is the LDAP server's TCP/IP hostname, and `[:port]` is the port number. If a port number is not specified, DB2 will use the default LDAP port (389).

  DB2 objects are located in the LDAP base distinguished name (baseDN). If you are using IBM SecureWay LDAP directory server Version 3.1, you do not have to configure the base distinguished name since DB2 can

dynamically obtain this information from the server. However, if you are using IBM eNetwork Directory Server Version 2.1, you must configure the LDAP base distinguished name on each machine by using the DB2SET command:

```
db2set DB2LDAP_BASEDN=<baseDN>
```

where `baseDN` is the name of the LDAP suffix that is defined at the LDAP server. This LDAP suffix is used to contain DB2 objects.

- The LDAP user's distinguished name (DN) and password. These are required only if you plan to use LDAP to store DB2 user-specific information.

Creating an LDAP User

DB2 supports setting DB2 registry variables and CLI configuration at the user level. (This is not available on the AIX and Solaris platforms.) User level support provides user-specific settings in a multi-user environment. An example is Windows NT Terminal Server where each logged on user can customize his or her own environment without interfering with the system environment or another user's environment.

When using the IBM LDAP directory, you must define an LDAP user before you can store user-level information in LDAP. You can create an LDAP user in one of the following ways:

- Create an LDIF file to contain all attributes for the user object, then run the LDIF import utility to import the object into the LDAP directory. The LDIF utility for the IBM LDAP server is "LDIF2DB".
- Use the Directory Management Tool (DMT), available only for the IBM SecureWay LDAP Directory Server Version 3.1, to create the user object.

A LDIF file containing the attributes for a person object appears similar to the following:

```
File name: newuser.ldif

dn: cn=Mary Burnnet, ou=DB2 UDB Development, ou=Toronto, o=ibm, c=ca
objectclass: ePerson
cn: Mary Burnnet
sn: Burnnet
uid: mburnnet
userPassword: password
telephonenumber: 1-416-123-4567
facsimiletelephonenumber: 1-416-123-4568
title: Software Developer
```

Following is an example of the LDIF command to import an LDIF file using the IBM LDIF import utility:

```
LDIF2DB -i newuser.ldif
```

Notes:

1. You must run the LDIF2DB command from the LDAP server machine.
2. You must grant the required access (ACL) to the LDAP user object so that the LDAP user can add, delete, read, and write to his own object. To grant ACL for the user object, use the LDAP Directory Server Web Administration tool.

Configuring the LDAP User for DB2 Applications

When working with the IBM LDAP client and before running DB2, you must configure the LDAP user distinguished name (DN) and password for the current logged on user. This can be done using the db2ldcfg utility:

```
db2ldcfg -u <userDN> -w <password> —> set the user's DN and password
         -r                        —> clear the user's DN and password
```

For example:

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 UDB Development,ou=Toronto,o=ibm,c=ca"
         -w password
```

Registration of DB2 Servers After Installation

Each DB2 server instance must be registered in LDAP to publish the protocol configuration information that is used by the client applications to connect to the DB2 server instance. When registering an instance of the database server, you need to specify a *node name*. The node name is used by client applications when they connect or attach to the server. You can catalog another alias name for the LDAP node by using the CATALOG LDAP NODE command.

Note: If you are working in a Windows 2000 domain environment, then during installation the DB2 server instance is automatically registered in the Active Directory with the following information:

```
nodename: TCP/IP hostname
protocol type: TCP/IP
```

If the TCP/IP hostname is longer than 8 characters, it will be truncated to 8 characters.

The REGISTER command appears as follows:

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
```

The `protocol` clause specifies the communication protocol to use when connecting to this database server.

When creating an instance for DB2 Universal Database EEE that includes multiple physical machines, the REGISTER command must be invoked once for each machine. Use the *rah* command to issue the REGISTER command on all machines.

Note: The same `ldap_node_name` cannot be used for each machine since the name must be unique in LDAP. You will want to substitute the hostname of each machine for the `ldap_node_name` in the REGISTER command. For example:

```
rah ">DB2 REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

The "<>" is substituted by the hostname on each machine where the `rah` command is run. In the rare occurrence where there are multiple DB2 Universal Database EEE instances, the combination of the instance and host index may be used as the node name in the *rah* command.

The REGISTER command can be issued for a remote DB2 server. To do so, you must specify the remote computer name, instance name, and the protocol configuration parameters when registering a remote server. The command can be used as follows:

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
   hostname <host_name>
   svcename <tcpip_service_name>
   remote <remote_computer_name>
   instance <instance_name>
```

The following convention is used for the computer name:

- If TCP/IP is configured, the computer name must be the same as the TCP/IP hostname.
- If APPN is configured, use the partner-LU name as the computer name.

When running in a high availability or failover environment, and using TCP/IP as the communication protocol, the *cluster* IP address must be used. Using the cluster IP address allows the client to connect to the server on either machine without having to catalog a separate TCP/IP node for each machine. The cluster IP address is specified using the `hostname` clause, shown as follows:

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
   hostname n.nn.nn.nn
```

where `n.nn.nn.nn` is the cluster IP address.

Refer to the *Command Reference* for additional information on the REGISTER command.

Update the Protocol Information for the DB2 Server

The DB2 server information in LDAP must be kept current. For example, changes to the protocol configuration parameters or the server network address require an update to LDAP.

To update the DB2 server in LDAP on the local machine, use the following command:

```
db2 update ldap ...
```

Examples of protocol configuration parameters that can be updated include:

- A TCP/IP hostname and service name or port number parameters.
- An IPX address.
- APPC protocol information like TP name, partner LU, or mode.
- A NetBIOS workstation name.

To update a remote DB2 server protocol configuration parameters use the UPDATE LDAP command with a `node` clause:

```
db2 update ldap
   node <node_name>
   hostname <host_name>
   svcename <tcpip_service_name>
```

Refer to the *Command Reference* for more information on the UPDATE LDAP command.

Catalog a Node Alias for ATTACH

A node name for the DB2 server must be specified when registering the server in LDAP. Applications use the node name to attach to the database server. If a different node name is required, such as when the node name is hard-coded in an application, use the CATALOG LDAP NODE command to make the change. The command would be similar to:

```
db2 catalog ldap node <ldap_node_name>
   as <new_alias_name>
```

To uncatalog a LDAP node, use the UNCATALOG LDAP NODE COMMAND. The command would appear similar to:

```
db2 uncatalog ldap node <ldap_node_name>
```

Deregistering the DB2 Server

Deregistration of an instance from LDAP also removes all the node, or alias, objects and the database objects referring to the instance.

Deregistration of the DB2 server on either a local or a remote machine requires the LDAP node name be specified for the server:

```
db2 deregister db2 server in ldap
   node <node_name>
```

When the DB2 server is deregistered, any LDAP node entry and LDAP database entries referring to the same instance of the DB2 server are also uncataloged.

Registration of Databases

During the creation of a database within an instance, the database is automatically registered in LDAP. Registration allows remote client connection to the database without having to catalog the database and node on the client machine. When a client attempts to connect to a database, if the database does not exist in the database directory on the local machine then the LDAP directory is searched.

If the name already exists in the LDAP directory, the database is still created on the local machine but a warning message is returned stating the naming conflict in the LDAP directory. For this reason you can manually catalog a database in the LDAP directory. The user can register databases on a remote server in LDAP by using the CATALOG LDAP DATABASE command. When registering a remote database, you specify the name of the LDAP node that represents the remote database server. You **must** register the remote database server in LDAP using the REGISTER DB2 SERVER IN LDAP command **before** registering the database.

To register a database manually in LDAP, use the CATALOG LDAP DATABASE command:

```
db2 catalog ldap database <dbname>
   at node <node_name>
   with "My LDAP database"
```

Attaching to a Remote Server

In the LDAP environment, you can attach to a remote database server using the LDAP node name on the ATTACH command:

```
db2 attach to <ldap_node_name>
```

When a client application attaches to a node or connects to a database for the first time, since the node is not in the local node directory, DB2 searches the LDAP directory for the target node entry. If the entry is found in the LDAP directory, the protocol information of the remote server is retrieved. If you connect to the database and if the entry is found in the LDAP directory, then the database information is also retrieved. Using this information, DB2 automatically catalogs a database entry and a node entry on the local machine. The next time the client application attaches to the same node or database, the information in the local database directory is used without having to search the LDAP directory.

In more detail: A caching mechanism exists so that the client only searches the LDAP directory once in its local directory catalogs. Once the information is retrieved, it is stored or cached on the local machine. Subsequent access to the same information is based on the values of the *dir_cache* database manager configuration parameter and the DB2LDAPCACHE registry variable.

- If DB2LDAPCACHE=NO and *dir_cache*=NO, then always read the information from LDAP.
- If DB2LDAPCACHE=NO and *dir_cache*=YES, then read the information from LDAP once and insert it into the DB2 cache.
- If DB2LDAPCACHE=YES or is not set, and if the required information is not found in the local cache, then the information is read from the LDAP directory and the local cache is refreshed.

Note: The caching of LDAP information is not applicable to user-level CLI or DB2 profile registry variables. Also, there is an "in-memory" cache for the database, node, and DCS directories. However, there is no such cache for just the node directory.

Deregistering the Database

The database is automatically deregistered from LDAP when:

- The database is dropped.
- The owning instance is deregistered from LDAP.

The database can be manually deregistered from LDAP using:

```
db2 uncatalog ldap database <dbname>
```

Refreshing LDAP Entries in Local Database and Node Directories

LDAP information is subject to change, so it is necessary to refresh the LDAP entries in the local and node directories. The local database and node directories are used to cache the entries in LDAP.

In more detail: A caching mechanism exists so that the client only searches the LDAP directory once in its local directory catalogs. Once the information is retrieved, it is stored or cached on the local machine. Subsequent access to the same information is based on the values of the *dir_cache* database manager configuration parameter and the DB2LDAPCACHE registry variable.

- If DB2LDAPCACHE=NO and *dir_cache*=NO, then always read the information from LDAP.
- If DB2LDAPCACHE=NO and *dir_cache*=YES, then read the information from LDAP once and insert it into the DB2 cache.
- If DB2LDAPCACHE=YES or is not set, and if the required information is not found in the local cache, then the information is read from the LDAP directory and the local cache is refreshed.

Note: The caching of LDAP information is not applicable to user-level CLI or DB2 profile registry variables. Also, there is an "in-memory" cache for the database, node, and DCS directories. However, there is no such cache for just the node directory.

To refresh the database entries that refer to LDAP resources, use the following command:

```
db2 refresh ldap database directory
```

To refresh the node entries on the local machine that refer to LDAP resources, use the following command:

```
db2 refresh ldap node directory
```

As part of the refresh, all the LDAP entries that are saved in the local database and node directories are removed. The next time that the application accesses the database or node, it will read the information directly from LDAP and generate a new entry in the local database or node directory.

To ensure the refresh is done in a timely way, you may want to:

- Schedule a refresh that is run periodically.
- Run the REFRESH command during system bootup.
- Use an available administration package to invoke the REFRESH command on all client machines.
- Set DB2LDAPCACHE="NO" to avoid LDAP information being cached in the database, node, and DCS directories.

Searching

DB2 searches the current LDAP directory partition, or current Active Directory domain in the Windows 2000 environment. In an environment where there are multiple LDAP directory partitions or domains, you can set the search scope. For example, if the information is not found in the current partition or domain, automatic search of all other partitions or domains can be requested. On the other hand, the search scope can be restricted to search only the local machine.

The search scope is controlled through the DB2 profile registry variable, DB2LDAP_SEARCH_SCOPE. To set the search scope value at the global level in LDAP, use the "-gl" option, which means "global in LDAP", on the *db2set* command:

```
db2set -gl db2ldap_search_scope=<value>
```

Possible values include: "local", "domain", or "global". The default value is "domain" which limits the search scope to the current directory partition. Setting the search scope in LDAP allows the setting of the default search scope for the entire enterprise. For example, you may want to initialize the search scope to "global" after a new database is created. This allows any client machine to search all other partitions or domains to find a database that is defined in a particular partition or domain. Once the entry has been recorded on each machine after the first connect or attach for each client, the search scope can be changed to "local". Once changed to "local", each client will not scan any partition or domain.

Note: The DB2 profile registry variable DB2LDAP_SEARCH_SCOPE is the only registry variable that supports setting the variable at the global level in LDAP.

Registering Host Databases

When registering host databases in LDAP, there are two possible configurations:

- Direct connection to the host databases; or,
- Connection to the host database though a gateway.

In the first case, the user would register the host server in LDAP, then catalog the host database in LDAP specifying the node name of the host server. In the second case, the user would register the gateway server in LDAP, then catalog the host database in LDAP specifying the node name of the gateway server.

As an example showing both cases, consider the following: Suppose there is a host database called NIAGARA_FALLS. It can accept incoming connections

using APPN and TCP/IP. If the client can not connect directly to the host because it does not have DB2 Connect, then it will connect using a gateway called "goto@niagara".

The following steps need to be done:

1. Register the host database server in LDAP for APPN connectivity. The REMOTE and INSTANCE clauses are arbitrary. The NODETYPE clause is set to "DCS" to indicate that this is a host database server.

   ```
   db2 register ldap as nfappn appn network CAIBMOML partnerlu NFLU
      mode IBMRDB remote mvssys instance msvinst nodetype dcs
   ```

2. Register the host database server in LDAP for TCP/IP connectivity. The TCP/IP hostname of the server is "myhost" and the port number is "446". Similar to step 1, the NODETYPE clause is set to "DCS" to indicate that this is a host database server.

   ```
   db2 register ldap as nftcpip tcpip hostname myhost svcename 446
      remote mvssys instance mvsinst nodetype dcs
   ```

3. Register a DB2 Connect gateway server in LDAP for TCP/IP connectivity. The TCP/IP hostname for the gateway server is "niagara" and the port number is "50000".

   ```
   db2 register ldap as whasf tcpip hostname niagara svcename 50000
      remote niagara instance goto nodetype server
   ```

4. Catalog the host database in LDAP using TCP/IP connectivity. The host database name is "NIAGARA_FALLS", the database alias name is "nftcpip". The GWNODE clause is used to specify the nodename of the DB2 Connect gateway server.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nftcpip at node nftcpip
      gwnode whasf authentication dcs
   ```

5. Catalog the host database in LDAP using APPN connectivity.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nfappn at node nfappn
      gwnode whasf authentication dcs
   ```

After completing the registration and cataloging shown above, if you want to connect to the host using TCPIP, you connect to "nftcpip". If you want to connect to the host using APPN, you connect to "nfappn". If you do not have DB2 Connect on your client workstation, the connection will go through the gateway using TCPIP and from there, depending on whether you use "nftcpip" or "nfappn", it will connect to host using TCP/IP or APPN respectively.

In general then, you can manually configure host database information in LDAP so that each client does not need to manually catalog the database and node locally on each machine. The process follows:

1. Register the host database server in LDAP. You must specify the remote computer name, instance name, and the node type for the host database

server in the REGISTER command using the REMOTE, INSTANCE, and NODETYPE clauses respectively. The REMOTE clause can be set to either the host name or the LU name of the host server machine. The INSTANCE clause can be set to any character string that has eight characters or less. (For example, the instance name can be set to "DB2".) The NODE TYPE clause must be set to "DCS" to indicate that this is a host database server.

2. Register the host database in LDAP using the CATALOG LDAP DATABASE command. Any additional DRDA parameters can be specified by using the PARMS clause. The database authentication type should be set to "DCS".

Setting DB2 Registry Variables at the User Level

Under the LDAP environment, the DB2 profile registry variables can be set at the user level which allows a user to customize their own DB2 environment. To set the DB2 profile registry variables at the user level, use the `-ul` option:

```
db2set -ul <variable>=<value>
```

Note: This is not supported on AIX or Solaris.

DB2 has a caching mechanism. The DB2 profile registry variables at the user level are cached on the local machine. If the `-ul` parameter is specified, DB2 always reads from the cache for the DB2 registry variables. The cache is refreshed when:

- You update or reset a DB2 registry variable at the user level.
- The command to refresh the LDAP profile variables at the user level is:

  ```
  db2set -ur
  ```

Enabling LDAP Support After Installation is Complete

To enable LDAP support at some point following the completion of the installation process, use the following procedure on each machine:

- Install the LDAP support binary files. Run the setup program and select the LDAP Directory Exploitation support from `Custom` install. The setup program installs the binary files and sets the DB2 profile registry variable DB2_ENABLE_LDAP to "YES".

  Note: For Windows 98/NT and UNIX platforms, you must explicitly enable LDAP by setting the DB2_ENABLE_LDAP registry variable to "YES" using the **db2set** command.

- (On UNIX platforms only) Declare the LDAP server's TCP/IP host name and (optional) port number using the following command:

  ```
  db2set DB2LDAPHOST=<base_domain_name>[:port_number]
  ```

where base_domain_name is the LDAP server's TCP/IP hostname, and [:port] is the port number. If a port number is not specified, DB2 will use the default LDAP port (389).

DB2 objects are located in the LDAP base distinguished name (baseDN). If you are using IBM SecureWay LDAP directory server Version 3.1, you do not have to configure the base distinguished name since DB2 can dynamically obtain this information from the server. However, if you are using IBM eNetwork Directory Server Version 2.1, you must configure the LDAP base distinguished name on each machine by using the DB2SET command:

```
db2set DB2LDAP_BASEDN=<baseDN>
```

where baseDN is the name of the LDAP suffix that is defined at the LDAP server. This LDAP suffix is used to contain DB2 objects.

- Register the current instance of the DB2 server in LDAP by using the REGISTER LDAP AS command. For example:

```
db2 register ldap as <node-name> protocol tcpip
```

- Run the CATALOG LDAP DATABASE command if you have databases you would like to register in LDAP. For example:

```
db2 catalog ldap database <dbname> as <alias_dbname>
```

- Enter the LDAP user's distinguished name (DN) and password. These are required only if you plan to use LDAP to store DB2 user-specific information.

Disabling LDAP Support

To disable LDAP support, use the following procedure:

- For each instance of the DB2 server, deregister the DB2 server from LDAP:

```
db2 deregister db2 server in ldap node <nodename>
```

- Set the DB2 profile registry variable DB2_ENABLE_LDAP to "NO".

LDAP Support and DB2 Connect

If LDAP support is available at the DB2 Connect gateway, and the database is not found at the gateway database directory, then DB2 will look up LDAP and attempt to keep the found information.

Security Considerations

Before accessing information in the LDAP directory, an application or user is authenticated by the LDAP server. The authentication process is called *binding* to the LDAP server.

It is important to apply access control on the information stored in the LDAP directory to prevent anonymous users from adding, deleting, or modifying the information.

Access control is inherited by default and can be applied at the container level. When a new object is created, it inherits the same security attribute as the parent object. An administration tool available for the LDAP server can be used to define access control for the container object.

By default, access control is defined as follows:

- For database and node entries in LDAP, everyone (or any anonymous user) has read access. Only the Directory Administrator and the owner or creator of the object has read/write access.
- For user profiles, the profile owner and the Directory Administrator have read/write access. One user cannot access the profile of another user if that user does not have Directory Administrator authority.

Note: The authorization check is always performed by the LDAP server and not by DB2. The LDAP authorization check is not related to DB2 authorization. An account or auth ID that has SYSADM authority may not have access to the LDAP directory.

When running the LDAP commands or APIs, if the bind Distinguished Name (bindDN) and password are not specified, DB2 binds to the LDAP server using the default credentials which may not have sufficient authority to perform the requested commands and an error will be returned.

You can explicitly specify the user's bindDN and password using the USER and PASSWORD clauses for the DB2 commands or APIs. Refer to the *Command Reference* for more information on DB2 commands, and to the *Administrative API Reference* for more information on DB2 APIs.

Security Considerations for Windows 2000 Active Directory

The DB2 database and node objects are created under the computer object of the machine where the DB2 server is installed in the Active Directory. To register a database server or catalog a database in the Active Directory, you need to have sufficient access to create and/or update the objects under the computer object.

By default, objects under the computer object are readable by any authenticated users and updateable by administrators (users that belong to the Administrators, Domain Administrators, and Enterprise Administrators groups). To grant access for a specific user or a group, use the *Active Directory Users and Computer* Management Console (MMC) as follows:

1. Start the *Active Directory Users and Computer* administration tool

(Start—> Program—> Administration Tools—> Active Directory Users and Computer)

2. Under *View*, select *Advanced Features*
3. Select the *Computers* container
4. Right click on the computer object that represents the server machine where DB2 is installed and select *Properties*
5. Select the *Security* tab, then add the required access to the specified user or group

The DB2 registry variables and CLI settings at the user level are maintained in the DB2 property object under the user object. To set the DB2 registry variables or CLI settings at the user level, a user needs to have sufficient access to create objects under the User object.

By default, only administrators have access to create objects under the User object. To grant access to a user to set the DB2 registry variables or CLI settings at the user level, use the *Active Directory Users and Computer* Management Console (MMC) as follows:

1. Start the *Active Directory Users and Computer* administration tool

   (Start—> Program—> Administration Tools—> Active Directory Users and Computer)
2. Select the user object under the Users container
3. Right click on the user object and select *Properties*
4. Select the *Security* tab
5. Add the user name to the list by using the Add button
6. Grant "Write", and "Create All Child Objects" access
7. Using the Advanced setting, set permissions to apply onto "This object and all child objects"
8. Select the check box "Allow inheritable permissions from parent to propagate to this object"

Extending the Directory Schema with DB2 Object Classes and Attributes

The LDAP Directory Schema defines object classes and attributes for the information stored in the LDAP directory entries. An object class consists of a set of mandatory and optional attributes. Every entry in the LDAP directory has an object class associated with it.

Before DB2 can store the information into LDAP, the Directory Schema for the LDAP server must include the object classes and attributes that DB2 uses. The process of adding new object classes and attributes to the base schema is called extending the Directory Schema.

Note: If you are using IBM SecureWay LDAP Directory v3.1, all the object classes and attributes that are required by DB2 are included in the base schema. You do not have to extend the base schema with DB2 object classes and attributes.

Extending the Directory Schema for IBM eNetwork Directory Version 2.1

When using the IBM eNetwork Directory Version 2.1, you must extend the base schema with the object classes and attributes that are used by DB2.

Use the following steps to extend the base schema for IBM eNetwork Directory Version 2.1:

1. Copy the DB2 attribute definition file, `db2.at`, and object class definition file, `db2.oc`, to the same directory that contains the system attribute and object class definition files, `slapd.at.conf` and `slapd.oc.conf`. The DB2 attribute and object class definition files can be found in the `cfg` subdirectory of the `sqllib` subdirectory. The system attribute and object class definition files are located in the `etc` subdirectory of the `%LDAPHome%` subdirectory.
2. Review the DB2 attribute and object class definition files. Comment out any object classes and attributes that have been defined in your current LDAP Directory Schema.
3. Add a line at the end of the `slapd.oc.conf` file as follows:
   ```
   include db2.oc
   ```
4. Add a line at the end of the `slapd.at.conf` file as follows:
   ```
   include db2.at
   ```
5. Restart the LDAP server.

Extending the Directory Schema for Windows 2000 Active Directory

Before DB2 can store information in the Windows 2000 Active Directory, the directory schema needs to be extended to include the new DB2 object classes and attributes. The process of adding new object classes and attributes to the directory schema is called *schema extension*.

You must extend the schema for Active Directory by running the DB2 Schema Installation program, **db2schex** before the first installation of DB2 on any machine that is part of a Windows 2000 domain.

The **db2schex** program is found on the product CD-ROM. The location of this program on the CD-ROM is under the `db2` directory and the `common` subdirectory. For example:

```
x:\db2\common
```

where `x:` is the CD-ROM drive.

The command is used as shown:

```
db2schex
```

There are other optional clauses associated with this command:

- –b UserDN

  To specify the user Distinguished Name.
- –w Password

  To specify the bind password.
- –u

  To uninstall the schema.
- –k

  To force uninstall to continue, ignoring errors.

Notes:

1. If no UserDN and password are specified, **db2schex** binds as the currently logged user.
2. The userDN clause can be specified as a Windows NT username.
3. To update the schema, you must be a member of the Schema Administrators group or have been delegated the rights to update the schema.

Examples:

- To install the DB2 schema:

  ```
  db2schex
  ```
- To install the DB2 schema and specify a bind DN and password:

  ```
  db2schex -b "cn=A Name,dc=toronto1,dc=ibm,dc=com"
           -w password
  ```

  Or,

  ```
  db2schex -b Administrator -w password
  ```
- To uninstall the DB2 schema:

  ```
  db2schex -u
  ```
- To uninstall the DB2 schema and ignore errors:

  ```
  db2schex -u -k
  ```

The DB2 Schema Installation program for Active Directory carries out the following tasks:

Notes:

1. Detects which server is the Schema Master
2. Binds to the Domain Controller that is the Schema Master
3. Ensures that the user has sufficient rights to add classes and attributes to the schema

4. Ensures that the schema master is writable (that is, the safety interlock in the registry is removed)
5. Creates all the new attributes
6. Creates all the new object classes
7. Detects errors, and if they occur, the program will roll back any changes to the schema.

DB2 Objects in the Windows 2000 Active Directory

DB2 creates objects in the Active Directory at two locations:

1. The DB2 database and node objects are created under the computer object of the machine where the DB2 Server is installed. For the DB2 server machine that does not belong to the Windows NT domain, the DB2 database and node objects are created under the "System" container.
2. The DB2 registry variables and CLI settings at the user level are stored in the DB2 property objects under the User object. These objects contain information that is specific to that user.

Object Classes and Attributes Used by DB2

The following tables describe the object classes that are used by DB2:

Table 27. cimManagedElement

Class	**cimManagedElement**
Active Directory LDAP Display Name	Not applicable
Active Directory Common Name (cn)	Not applicable
Description	Provides a base class of many of the system management object classes in the IBM Schema
SubClassOf	top
Required Attribute(s)	
Optional Attribute(s)	description
Type	abstract
OID (Object Identifier)	1.3.18.0.2.6.132
GUID (Global Unique Identifier)	b3afd63f-5c5b-11d3-b818-002035559151

Table 28. cimSetting

Class	**cimSetting**
Active Directory LDAP Display Name	Not applicable
Active Directory Common Name (cn)	Not applicable
Description	Provides a base class for configuration and settings in the IBM Schema
SubClassOf	cimManagedElement

Table 28. cimSetting (continued)

Class	**cimSetting**
Required Attribute(s)	
Optional Attribute(s)	settingID
Type	abstract
OID (object identifier)	1.3.18.0.2.6.131
GUID (Global Unique Identifier)	b3afd64d-5c5b-11d3-b818-002035559151

Table 29. eProperty

Class	**eProperty**
Active Directory LDAP Display Name	ibm-eProperty
Active Directory Common Name (cn)	ibm-eProperty
Description	Used to specify any application specific settings for user preference properties
SubClassOf	cimSetting
Required Attribute(s)	
Optional Attribute(s)	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
Type	structural
OID (object identifier)	1.3.18.0.2.6.90
GUID (Global Unique Identifier)	b3afd69c-5c5b-11d3-b818-002035559151

Table 30. DB2Node

Class	**DB2Node**
Active Directory LDAP Display Name	ibm-db2Node
Active Directory Common Name (cn)	ibm-db2Node
Description	Represents a DB2 Server
SubClassOf	eSap / ServiceConnectionPoint

Table 30. DB2Node (continued)

Class	**DB2Node**
Required Attribute(s)	db2nodeName
Optional Attribute(s)	db2nodeAlias db2instanceName db2Type host / dNSHostName (see Note 2) protocolInformation/ServiceBindingInformation
Type	structural
OID (object identifier)	1.3.18.0.2.6.116
GUID (Global Unique Identifier)	b3afd65a-5c5b-11d3-b818-002035559151
Special Notes	1. The *DB2Node* class is derived from *eSap* object class under IBM SecureWay directory and from *ServiceConnectionPoint* object class under Microsoft Active Directory. 2. The *host* is used under IBM SecureWay environment. The *dNSHostName* attribute is used under Microsoft Active Directory. 3. The *protocolInformation* is only used under IBM SecureWay environment. For Microsoft Active Directory, the attribute *ServiceBindingInformation*, inherited from the ServiceConnectionPoint class, is used to contain the protocol information.

The *protocolInformation* (in IBM SecureWay Directory) or *ServiceBindingInformation* (in Microsoft Active Directory) attribute in the *DB2Node* object contains the communication protocol information to bind the DB2 database server. It consists of tokens that describe the network protocol supported. Each token is separated by a semicolon. There is no space between the tokens. An asterisk (*) may be used to specify an optional parameter.

The tokens for TCP/IP are:

- "TCPIP"
- Server hostname or IP address
- Service name (svcename) or port number (e.g. 50000)
- (Optional) security ("NONE" or "SOCKS")

The tokens for APPN are:

- "APPN"
- Network ID
- Partner LU
- Transaction Program (TP) Name (Support Application TP only, does not support Service TP – TP in HEX)
- Mode
- Security (either "NONE", "SAME", or "PROGRAM")
- (Optional) LAN adapter address
- (Optional) Change password LU

Note: On a DB2 for Windows NT client (or for Windows 98), if the APPN information is not configured on the local SNA stack; and, if the LAN adapter address and optional change password LU are found in LDAP, then the DB2 client tries to use this information to configure the SNA stack if it knows how to configure the stack. This support is not available on DB2 for AIX, or DB2 for Solaris, clients.

The tokens for IPX/SPX are:
- "IPXSPX"
- IPX address

The IPX/SPX listener is available on the DB2 server (not on the client) for AIX and Solaris. NetBIOS and NPIPE are not supported on AIX and Solaris.

The tokens for NetBIOS are:
- "NETBIOS"
- Server NetBIOS workstation name

The tokens for Named Pipe are:
- "NPIPE"
- Computer name of the server
- Instance name of the server

Table 31. DB2Database

Class	**DB2Database**
Active Directory LDAP Display Name	ibm-db2Database
Active Directory Common Name (cn)	ibm-db2Database
Description	Represents a DB2 database
SubClassOf	top

Table 31. DB2Database (continued)

Class	**DB2Database**
Required Attribute(s)	db2databaseName db2nodePtr
Optional Attribute(s)	db2databaseAlias db2additionalParameter db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName
Type	structural
OID (object identifier)	1.3.18.0.2.6.117
GUID (Global Unique Identifier)	b3afd659-5c5b-11d3-b818-002035559151

Table 32. db2additionalParameters

Attribute	**db2additionalParameters**
Active Directory LDAP Display Name	ibm-db2AdditionalParameters
Active Directory Common Name (cn)	ibm-db2AdditionalParameters
Description	Contains any additional parameters used when connecting to the host database server
Syntax	Case Ignore String
Maximum Length	1024
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.426
GUID (Global Unique Identifier)	b3afd315-5c5b-11d3-b818-002035559151

Table 33. db2authenticationLocation

Attribute	**db2authenticationLocation**
Active Directory LDAP Display Name	ibm-db2AuthenticationLocation
Active Directory Common Name (cn)	ibm-db2AuthenticationLocation
Description	Specifies where authentication takes place
Syntax	Case Ignore String

Table 33. db2authenticationLocation (continued)

Attribute	db2authenticationLocation
Maximum Length	64
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.425
GUID (Global Unique Identifier)	b3afd317-5c5b-11d3-b818-002035559151
Notes	Valid values are: CLIENT, SERVER, DCS, DCE, KERBEROS, SVRENCRYPT, or DCSENCRYPT

Table 34. db2ARLibrary

Attribute	db2ARLibrary
Active Directory LDAP Display Name	ibm-db2ARLibrary
Active Directory Common Name (cn)	ibm-db2ARLibrary
Description	Name of the Application Requestor library
Syntax	Case Ignore String
Maximum Length	256
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.427
GUID (Global Unique Identifier)	b3afd316-5c5b-11d3-b818-002035559151

Table 35. db2databaseAlias

Attribute	db2databaseAlias
Active Directory LDAP Display Name	ibm-db2DatabaseAlias
Active Directory Common Name (cn)	ibm-db2DatabaseAlias
Description	Database alias name(s)
Syntax	Case Ignore String
Maximum Length	1024
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.422
GUID (Global Unique Identifier)	b3afd318-5c5b-11d3-b818-002035559151

Table 36. db2databaseName

Attribute	db2databaseName
Active Directory LDAP Display Name	ibm-db2DatabaseName
Active Directory Common Name (cn)	ibm-db2DatabaseName

Table 36. db2databaseName (continued)

Attribute	**db2databaseName**
Description	Database name
Syntax	Case Ignore String
Maximum Length	1024
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.421
GUID (Global Unique Identifier)	b3afd319-5c5b-11d3-b818-002035559151

Table 37. db2databaseRelease

Attribute	**db2databaseRelease**
Active Directory LDAP Display Name	ibm-db2DatabaseRelease
Active Directory Common Name (cn)	ibm-db2DatabaseRelease
Description	Database release number
Syntax	Case Ignore String
Maximum Length	64
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.429
GUID (Global Unique Identifier)	b3afd31a-5c5b-11d3-b818-002035559151

Table 38. db2nodeAlias

Attribute	**db2nodeAlias**
Active Directory LDAP Display Name	ibm-db2NodeAlias
Active Directory Common Name (cn)	ibm-db2NodeAlias
Description	Node alias name(s)
Syntax	Case Ignore String
Maximum Length	1024
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.420
GUID (Global Unique Identifier)	b3afd31d-5c5b-11d3-b818-002035559151

Table 39. db2nodeName

Attribute	**db2nodeName**
Active Directory LDAP Display Name	ibm-db2NodeName
Active Directory Common Name (cn)	ibm-db2NodeName

Table 39. db2nodeName (continued)

Attribute	**db2nodeName**
Description	Node name
Syntax	Case Ignore String
Maximum Length	64
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.419
GUID (Global Unique Identifier)	b3afd31e-5c5b-11d3-b818-002035559151

Table 40. db2nodePtr

Attribute	**db2nodePtr**
Active Directory LDAP Display Name	ibm-db2NodePtr
Active Directory Common Name (cn)	ibm-db2NodePtr
Description	Pointer to the Node (DB2Node) object that represents the database server which owns the database
Syntax	Distinguished Name
Maximum Length	1000
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.423
GUID (Global Unique Identifier)	b3afd31f-5c5b-11d3-b818-002035559151
Special Notes	This relationship allows the client to retrieve protocol communication information to connect to the database

Table 41. db2gwPtr

Attribute	**db2gwPtr**
Active Directory LDAP Display Name	ibm-db2GwPtr
Active Directory Common Name (cn)	ibm-db2GwPtr
Description	Pointer to the Node object that represents the gateway server and from which the database can be accessed
Syntax	Distinguished Name
Maximum Length	1000
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.424

Table 41. db2gwPtr (continued)

Attribute	db2gwPtr
GUID (Global Unique Identifier)	b3afd31b-5c5b-11d3-b818-002035559151

Table 42. db2instanceName

Attribute	db2instanceName
Active Directory LDAP Display Name	ibm-db2InstanceName
Active Directory Common Name (cn)	ibm-db2InstanceName
Description	The name of the database server instance
Syntax	Case Ignore String
Maximum Length	256
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.428
GUID (Global Unique Identifier)	b3afd31c-5c5b-11d3-b818-002035559151

Table 43. db2Type

Attribute	db2Type
Active Directory LDAP Display Name	ibm-db2Type
Active Directory Common Name (cn)	ibm-db2Type
Description	Type of the database server
Syntax	Case Ignore String
Maximum Length	64
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.418
GUID (Global Unique Identifier)	b3afd320-5c5b-11d3-b818-002035559151
Notes	Valid types for database server are: SERVER, MPP, and DCS

Table 44. DCEPrincipalName

Attribute	DCEPrincipalName
Active Directory LDAP Display Name	ibm-DCEPrincipalName
Active Directory Common Name (cn)	ibm-DCEPrincipalName
Description	DCE principal name
Syntax	Case Ignore String
Maximum Length	2048

Table 44. DCEPrincipalName (continued)

Attribute	**DCEPrincipalName**
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.443
GUID (Global Unique Identifier)	b3afd32d-5c5b-11d3-b818-002035559151

Table 45. cesProperty

Attribute	**cesProperty**
Active Directory LDAP Display Name	ibm-cesProperty
Active Directory Common Name (cn)	ibm-cesProperty
Description	Values of this attribute may be used to provide application-specific preference configuration parameters. For example, a value may contain XML-formatted data. All values of this attribute must be homogeneous in the cesPropertyType attribute value.
Syntax	Case Exact String
Maximum Length	32700
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.307
GUID (Global Unique Identifier)	b3afd2d5-5c5b-11d3-b818-002035559151

Table 46. cesPropertyType

Attribute	**cesPropertyType**
Active Directory LDAP Display Name	ibm-cesPropertyType
Active Directory Common Name (cn)	ibm-cesPropertyType
Description	Values of this attribute may be used to describe the syntax, semantics, or other characteristics of all of the values of the cesProperty attribute. For example, a value of “XML” might be used to indicate that all the values of the cesProperty attribute are encoded as XML syntax.
Syntax	Case Ignore String
Maximum Length	128
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.308
GUID (Global Unique Identifier)	b3afd2d6-5c5b-11d3-b818-002035559151

Table 47. cisProperty

Attribute	**cisProperty**
Active Directory LDAP Display Name	ibm-cisProperty
Active Directory Common Name (cn)	ibm-cisProperty
Description	Values of this attribute may be used to provide application-specific preference configuration parameters. For example, a value may contain an INI file. All values of this attribute must be homogeneous in their cisPropertyType attribute value.
Syntax	Case Ignore String
Maximum Length	32700
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.309
GUID (Global Unique Identifier)	b3afd2e0-5c5b-11d3-b818-002035559151

Table 48. cisPropertyType

Attribute	**cisPropertyType**
Active Directory LDAP Display Name	ibm-cisPropertyType
Active Directory Common Name (cn)	ibm-cisPropertyType
Description	Values of this attribute may be used to describe the syntax, semantics, or other characteristics of all of the values of the cisProperty attribute. For example, a value of “INI File” might be used to indicate that all the values of the cisProperty attribute are INI files.
Syntax	Case Ignore String
Maximum Length	128
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.310
GUID (Global Unique Identifier)	b3afd2e1-5c5b-11d3-b818-002035559151

Table 49. binProperty

Attribute	**binProperty**
Active Directory LDAP Display Name	ibm-binProperty
Active Directory Common Name (cn)	ibm-binProperty

Table 49. binProperty (continued)

Attribute	**binProperty**
Description	Values of this attribute may be used to provide application-specific preference configuration parameters. For example, a value may contain a set of binary-encoded Lotus 123 properties. All values of this attribute must be homogeneous in their binPropertyType attribute values.
Syntax	binary
Maximum Length	250000
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.305
GUID (Global Unique Identifier)	b3afd2ba-5c5b-11d3-b818-002035559151

Table 50. binPropertyType

Attribute	**binPropertyType**
Active Directory LDAP Display Name	ibm-binPropertyType
Active Directory Common Name (cn)	ibm-binPropertyType
Description	Values of this attribute may be used to describe the syntax, semantics, or other characteristics of all of the values of the binProperty attribute. For example, a value of "Lotus 123" might be used to indicate that all the values of the binProperty attribute are binary-encoded Lotus 123 properties.
Syntax	Case Ignore String
Maximum Length	128
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.306
GUID (Global Unique Identifier)	b3afd2bb-5c5b-11d3-b818-002035559151

Table 51. PropertyType

Attribute	**PropertyType**
Active Directory LDAP Display Name	ibm-propertyType
Active Directory Common Name (cn)	ibm-propertyType
Description	Values of this attribute describe the semantic characteristics of the eProperty object
Syntax	Case Ignore String
Maximum Length	128

Table 51. PropertyType (continued)

Attribute	PropertyType
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.320
GUID (Global Unique Identifier)	b3afd4ed-5c5b-11d3-b818-002035559151

Table 52. settingID

Attribute	settingID
Active Directory LDAP Display Name	Not applicable
Active Directory Common Name (cn)	Not applicable
Description	A naming attribute that may be used to identify the cimSetting derived object entries such as eProperty
Syntax	Case Ignore String
Maximum Length	256
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.325
GUID (Global Unique Identifier)	b3afd596-5c5b-11d3-b818-002035559151

Appendix K. Extending the Control Center

In Version 7, you can extend the DB2 Universal Database Control Center by using the new *plug-in* architecture to provide additional function.

The concept of the *plug-in* architecture is to provide the ability to add items for a given object in the Control Center popup menu, and add new buttons to the tool bar. A set of Java interfaces, which you must implement, is shipped along with the tools. These interfaces are used to communicate to the Control Center what additional actions to include.

Performance Considerations

The plug-in extensions (db2plug.zip) are loaded at the startup time of the Control Center tools. This may increase the startup time of the tools, depending on the size of the ZIP file; however, we expect that the plug-in ZIP file will be small for most users and the impact should be minimal.

Packaging Considerations

You must ZIP the extension class files according to the rules of a Java archive file. To run the Control Center tools as an application, the ZIP file (db2plug.zip) must be in the *classpath*. To run the Control Center tools as an applet, the ZIP file must be located where the <codebase> tag points to in the Control Center html file.

The ZIP file should be built with no compression and maintain the relative path positions of all the class files (`zip -r0 db2plug.zip *.class`).

Interface Descriptions

The following interfaces are shipped:

- CCExtension
- CCObject
- CCMenuAction
- CCToolbarAction.

The interfaces are described in the next sections, followed by an example.

CCExtension

The CCExtension interface allows you to extend the Control Center user interface by adding new toolbar buttons, new menu items, and overriding existing menu actions.

The external interface is defined as follows:

```
public interface CCExtension
{
   /**
    * Get an array of CCObject subclass objects which define
    * a list of objects to be inserted or overridden in the
    * Control Center
    * @return CCObject[] CCObject subclass objects array
    */
   public CCObject[] getObjects();

   /**
    * Get an array of CCToolbarAction subclass objects which represent
    * a list of buttons to be added to the Control Center
    * main toolbar.
    * @return CCToolbarAction[] CCToolbarAction subclass objects array
    */
   public CCToolbarAction[] getToolbarActions();
}
```

To use CCExtension, create a Java class which imports the "com.ibm.db2.tools.cc.navigator" package and implements this interface. The new class must provide the implementation of the getObjects() and getToolbarActions() methods.

The getObjects() method returns an array of CCObject which defines the existing objects to which the user would like to add new menu actions or remove a predefined set of menu actions.

The getToolbarActions() method returns an array of CCToolbarAction which will be added to the Control Center main toolbar.

A single CCExtension subclass file or multiple CCExtension subclass files can be used to define the Control Center extensions. For the Control Center to use these extensions, use the following setup procedure:

1. Create a "db2plug.zip" file which contains all the CCExtension subclass files. The files should not be compressed. For example, if the CCExtension files are in the plugin package and they are located in the plugin directory, use the following command:

   ```
   zip -r0 db2plug.zip plugin\*.class
   ```

   This command will put all the plugin package class files into the db2plug.zip file and preserve their relative path information.

2. To run the Control Center as an applet, put the db2plug.zip file in where the <codebase> tag points to in the Control Center html file. To run the Control Center as an application, put the db2plug.zip in a directory pointed to by the CLASSPATH environment variable.

For browsers that support multiple archives, just add "db2plug.zip" to the archive list of the Control Center html page. Otherwise, all the CCExtension, CCObject, CCToolbarAction, and CCMenuAction subclass files will have to be in their relative directories depending on which package they belong to.

CCObject

The CCObject interface allows you to change the behavior of the menu actions of an existing object.

The external interface is defined as follows:

```
public interface CCObject
{
   /**
    * The following static constants defines a list of object type
    * available to be added to the Control Center tree.
    */
   public static final int UDB_SYSTEMS_FOLDER                          = 0;
   public static final int UDB_SYSTEM                                  = 1;
   public static final int UDB_INSTANCES_FOLDER                        = 2;
   public static final int UDB_INSTANCE                                = 3;
   public static final int UDB_DATABASES_FOLDER                        = 4;
   public static final int UDB_DATABASE                                = 5;
   public static final int UDB_TABLES_FOLDER                           = 6;
   public static final int UDB_TABLE                                   = 7;
   public static final int UDB_TABLESPACES_FOLDER                      = 8;
   public static final int UDB_TABLESPACE                              = 9;
   public static final int UDB_VIEWS_FOLDER                            = 10;
   public static final int UDB_VIEW                                    = 11;
   public static final int UDB_ALIASES_FOLDER                          = 12;
   public static final int UDB_ALIAS                                   = 13;
   public static final int UDB_TRIGGERS_FOLDER                         = 14;
   public static final int UDB_TRIGGER                                 = 15;
   public static final int UDB_SCHEMAS_FOLDER                          = 16;
   public static final int UDB_SCHEMA                                  = 17;
   public static final int UDB_INDEXES_FOLDER                          = 18;
   public static final int UDB_INDEX                                   = 19;
   public static final int UDB_CONNECTIONS_FOLDER                      = 20;
   public static final int UDB_CONNECTION                              = 21;
   public static final int UDB_REPLICATION_SOURCES_FOLDER              = 22;
   public static final int UDB_REPLICATION_SOURCE                      = 23;
   public static final int UDB_REPLICATION_SUBSCRIPTIONS_FOLDER        = 24;
   public static final int UDB_REPLICATION_SUBSCRIPTION                = 25;
   public static final int UDB_BUFFERPOOLS_FOLDER                      = 26;
   public static final int UDB_BUFFERPOOL                              = 27;
   public static final int UDB_APPLICATION_OBJECTS_FOLDER              = 28;
   public static final int UDB_USER_DEFINED_DISTINCT_DATATYPES_FOLDER  = 29;
   public static final int UDB_USER_DEFINED_DISTINCT_DATATYPE          = 30;
   public static final int UDB_USER_DEFINED_DISTINCT_FUNCTIONS_FOLDER  = 31;
```

```
public static final int UDB_USER_DEFINED_DISTINCT_FUNCTION          = 32;
public static final int UDB_PACKAGES_FOLDER                         = 33;
public static final int UDB_PACKAGE                                 = 34;
public static final int UDB_STORE_PROCEDURES_FOLDER                 = 35;
public static final int UDB_STORE_PROCEDURE                         = 36;
public static final int UDB_USER_AND_GROUP_OBJECTS_FOLDER           = 37;
public static final int UDB_DB_USERS_FOLDER                         = 38;
public static final int UDB_DB_USER                                 = 39;
public static final int UDB_DB_GROUPS_FOLDER                        = 40;
public static final int UDB_DB_GROUP                                = 41;
public static final int UDB_DRDA_TABLE                              = 42;

public static final int S390_SUBSYSTEMS_FOLDER                      = 43;
public static final int S390_SUBSYSTEM                              = 44;
public static final int S390_BUFFERPOOLS_FOLDER                     = 45;
public static final int S390_BUFFERPOOL                             = 46;
public static final int S390_VIEWS_FOLDER                           = 47;
public static final int S390_VIEW                                   = 48;
public static final int S390_DATABASES_FOLDER                       = 49;
public static final int S390_DATABASE                               = 50;
public static final int S390_TABLESPACES_FOLDER                     = 51;
public static final int S390_TABLESPACE                             = 52;
public static final int S390_TABLES_FOLDER                          = 53;
public static final int S390_TABLE                                  = 54;
public static final int S390_INDEXS_FOLDER                          = 55;
public static final int S390_INDEX                                  = 56;
public static final int S390_STORAGE_GROUPS_FOLDER                  = 57;
public static final int S390_STORAGE_GROUP                          = 58;
public static final int S390_ALIASES_FOLDER                         = 59;
public static final int S390_ALIAS                                  = 60;
public static final int S390_SYNONYMS_FOLDER                        = 61;
public static final int S390_SYNONYM                                = 62;
public static final int S390_APPLICATION_OBJECTS_FOLDER             = 63;
public static final int S390_COLLECTIONS_FOLDER                     = 64;
public static final int S390_COLLECTION                             = 65;
public static final int S390_PACKAGES_FOLDER                        = 66;
public static final int S390_PACKAGE                                = 67;
public static final int S390_PLANS_FOLDER                           = 68;
public static final int S390_PLAN                                   = 69;
public static final int S390_PROCEDURES_FOLDER                      = 70;
public static final int S390_PROCEDURE                              = 71;
public static final int S390_DB_USERS_FOLDER                        = 72;
public static final int S390_DB_USER                                = 73;
public static final int S390_LOCATIONS_FOLDER                       = 74;
public static final int S390_LOCATION                               = 75;
public static final int S390_DISTINCT_TYPES_FOLDER                  = 76;
public static final int S390_DISTINCT_TYPE                          = 77;
public static final int S390_USER_DEFINED_FUNCTIONS_FOLDER          = 78;
public static final int S390_USER_DEFINED_FUNCTION                  = 79;
public static final int S390_TRIGGERS_FOLDER                        = 80;
public static final int S390_TRIGGER                                = 81;
public static final int S390_SCHEMAS_FOLDER                         = 82;
public static final int S390_SCHEMA                                 = 83;
public static final int S390_CATALOG_TABLES_FOLDER                  = 84;
public static final int S390_CATALOG_TABLE                          = 85;
```

```
public static final int DCS_GATEWAY_CONNECTIONS_FOLDER              = 86;
public static final int DCS_GATEWAY_CONNECTION                      = 87;

/**
 * Total number of object types
 */
public static final int NUM_OBJECT_TYPES                            = 88;

/**
 * Get the name of these object
 * The function returns the name of this object. This name
 * can be of three types:
 * (1) Fully qualified name
 *     Syntax: xxxxx-yyyyy-zzzzz
 *             where xxxxx-yyyyy is the fully quality name of the
 *             parent object and zzzzz is the name of the new object.
 *     Note: Parent and child object name is separated by '-' character.
 *     If a schema name is required to identify object, the fully
 *     qualified name is represented by xxxxx-yyyyy-wwwww.zzzzz
 *     where wwwww is the schema name.
 *     Only the behavior of the object that matches this fully
 *     qualified name will be affected.
 * (2) Parent fully qualified name
 *     Syntax: xxxxx-yyyyy
 *             where xxxxx-yyyyy is the fully qualified name of the
 *             parent object.
 *     When the object type is folder (ie. DATABASES_FOLDER), the
 *     getName() should only return the fully qualified name of the
 *     folder's parent.
 *     Only the behavior of the object that match this name
 *     and the specific type return by the getType() function will be
 *     affected.
 * (3) null
 *     Syntax: null
 *     If null is return, the CCActions returns by the getActions()
 *     call will be applied to all objects of type returns by the
 *     getType() call.
 * @return String object name
 */
public String getName();

/**
 * Get the type of this object
 * @return int return one of the static type constants defined
 * in this interface
 */
public int getType();

/**
 * Get the CCMenu Action array which defines the list of menu actions
 * to be created for the selected object
 * return CCMenuAction[] CCMenuAction array
 */
public CCMenuAction[] getMenuActions();
```

```
   /**
    * Check if this object is editable.
    * If not, the Alter related menu items will be removed from
    * the object's popup menu return boolean If false, the Alter
    * menu item will be removed from the object's popup menu
    */
   public boolean isEditable();

   /**
    * Check if this object is configurable.
    * If not, the configuration related menu items will be
    * removed from the object's popup menu return boolean If
    * false, the Configuration related menu item will be removed
    * from the object's popup menu
    */
   public boolean isConfigurable();
}
```

Note: At this time, the last two methods in `CCObject`: `isEditable()` and `isConfigurable()` should always return **true**.

CCMenuAction

The CCMenuAction interface allows you to define a new action to be used by a Control Center object.

The external interface is defined as follows:

```
public interface CCMenuAction
{
   /**
    * Get the name of this action
    * @return String Name text on the menu item
    */
   public String getMenuText();

   /**
    * Invoked when an action occurs. Use the getActionCommand()
    * method of the ActionEvent to get the fully qualified name of
    * the invoked Control Center object.
    * @param e Action event
    */
   public void actionPerformed(ActionEvent e);
}
```

CCToolBarAction

The CCToolbarAction interface allows you to define a new action on the Control Center toolbar.

The external interface is defined as follows:

```
public interface CCToolbarAction
{
   /**
```

```
     * Get the name of this action
     * @return String Name text on the menu item, or toolbar
     * button hover help
     */
    public String getHoverHelpText();

    /**
     * Get the icon for the toolbar button
     * Any toolbar CCAction should implement this function and return
     * a valid ImageIcon object. Otherwise, the button will have no icon.
     * @return ImageIcon Icon to be displayed
     */
    public ImageIcon getIcon();

    /**
     * Invoked when an action occurs.
     * @param e Action event
     */
    public void actionPerformed(ActionEvent e);
}
```

Usage Scenario

The code in the following example will:

1. Update the actions of the SAMPLE Database (see "MySample.java" on page 434)
2. Update the actions of all Database objects (see "MyDatabaseActions.java" on page 435)
3. Add a new instance object (see "MyInstance.java" on page 435)
4. Update the actions of the DB2 instance (see "MyDB2.java" on page 436)
5. Update the actions of the Databases folder (see "MyDatabases.java" on page 437)
6. Update the actions of the SYSIBM.SYSPLAN table (see "MySYSPLAN.java" on page 437)
7. Add a new table object (see "MyTable.java" on page 438)
8. Update the actions of the DB_User object under the Application object (see "MyDBUser.java" on page 439)
9. Add a button to the Control Center toolbar (see "MyToolbarAction.java" on page 439).

The main extension file is MyExtension.java. All the class files are stored in the plugin directory and are zipped up by the command:

```
zip -r0 db2plug.zip plugin
```

The output db2plug.zip is then placed in the CLASSPATH or in the codebase directory depending whether the Control Center is running as an application or an applet.

MyExtension.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyExtension implements CCExtension
{
   public CCObject[] getObjects()
   {
      CCObject[] objs = new CCObject[10];
      objs[0] = new MySample();
      objs[1] = new MyDatabaseActions();
      objs[2] = new MyInstance();
      objs[3] = new MyDB2();
      objs[4] = new MyDatabases();
      objs[5] = new MySYSPLAN();
      objs[6] = new MyTable();
      objs[7] = new MyDBUser();
      return objs;
   }

   public CCAction[] getActions()
   {
      CCAction[] actions = new CCAction[1];
      actions[0] = new MyToolbarAction();
      return actions;
   }
}
```

MySample.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MySample implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - SAMPLE";
   }

   public int getType()
   {
      return DATABASE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }
```

```
   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[2];
      acts[0] = new MyAlterAction();
      acts[1] = new MyAction();
      return acts;
   }

}
```

MyDatabaseActions.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDatabaseActions implements CCObject
{
   public String getName()
   {
      return null;
   }

   public int getType()
   {
      return DATABASE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[2];
      acts[0] = new MyDropAction();
      acts[1] = new MyAction();
      return acts;
   }

}
```

MyInstance.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyInstance implements CCObject
{
   public String getName()
   {
      return "LOCAL - MyInstance";
```

```
    }

    public int getType()
    {
       return INSTANCE;
    }

    public javax.swing.ImageIcon getIcon()
    {
       return null;
    }

    public boolean isNew()
    {
       return true;
    }

    public CCAction[] getActions()
    {
       CCAction[] acts = new CCAction[2];
       acts[0] = new MyAlterAction();
       acts[1] = new MyAction();
       return null;
    }

}
```

MyDB2.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDB2 implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2";
   }

   public int getType()
   {
      return INSTANCE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
```

```
        CCAction[] acts = new CCAction[3];
        acts[0] = new MyAlterAction();
        acts[1] = new MyAction();
        acts[2] = new MyCascadeAction();
        return acts;
    }
}
```

MyDatabases.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDatabases implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - Databases";
   }

   public int getType()
   {
      return DATABASE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[1];
      acts[0] = new MyCreateAction();
      return acts;
   }

}
```

MySYSPLAN.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MySYSPLAN implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - SAMPLE - SYSIBM - SYSPLAN";
   }

   public int getType()
```

```
    {
       return TABLE;
    }

    public javax.swing.ImageIcon getIcon()
    {
       return null;
    }

    public boolean isNew()
    {
       return false;
    }

    public CCAction[] getActions()
    {
       CCAction[] acts = new CCAction[2];
       acts[0] = new MyAlterAction();
       acts[1] = new MyAction();
       return acts;
    }

}
```

MyTable.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyTable implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - SAMPLE - SYSIBM - MyTable";
   }

   public int getType()
   {
      return TABLE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return true;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[2];
      acts[0] = new MyAlterAction();
      acts[1] = new MyAction();
```

```
        return acts;
    }

}
```

MyDBUser.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDBUser implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - TEST-DB Users";
   }

   public int getType()
   {
      return DB_USER;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[2];
      acts[0] = new MyAlterAction();
      acts[1] = new MyAction();
      return acts;
   }

}
```

MyToolbarAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;
import javax.swing.*;

public class MyToolbarAction extends CCAction
{
   public MyToolbarAction()
   {
      super("MyToolbarAction");
   }

   public ImageIcon getIcon()
   {
```

```
        return <Your icon>;
    }

    public boolean actionPerformed(String objectName)
    {
        System.out.println( "My action performed, object name = " +
                            objectName );
        return true;
    }
}
```

MyAlterAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyAlterAction extends CCAction
{
    public MyAlterAction()
    {
        super(0);
    }

    public boolean actionPerformed(String objectName)
    {
        System.out.println( "My alter action performed, object name = " +
                            objectName );
        return true;
    }
}
```

MyAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyAction extends CCAction
{
    public MyAction()
    {
        super("MyAction");
    }

    public boolean actionPerformed(String objectName)
    {
        System.out.println( "My action performed, object name = " +
                            objectName );
        return true;
    }
}
```

MyDropAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDropAction extends CCAction
```

```
{
   public MyDropAction()
   {
      super(1);
   }

   public boolean actionPerformed(String objectName)
   {
      System.out.println( "My drop action performed, object name = " +
                          objectName );
      return true;
   }
}
```

MyCascadeAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyCascadeAction extends CCAction
{
   public MyCascadeAction()
   {
      super(11,2);
   }

   public boolean actionPerformed(String objectName)
   {
      System.out.println( "My cascade action performed, object name = " +
                          objectName );
      return true;
   }
}
```

MyCreateAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyCreateAction extends CCAction
{
   public MyCreateAction()
   {
      super(0);
   }

   public boolean actionPerformed(String objectName)
   {
      System.out.println( "My create action performed, object name = " +
                          objectName );
      return true;
   }
}
```

Appendix L. Using the DB2 Library

The DB2 Universal Database library consists of online help, books (PDF and HTML), and sample programs in HTML format. This section describes the information that is provided, and how you can access it.

To access product information online, you can use the Information Center. For more information, see "Accessing Information with the Information Center" on page 457. You can view task information, DB2 books, troubleshooting information, sample programs, and DB2 information on the Web.

DB2 PDF Files and Printed Books

DB2 Information

The following table divides the DB2 books into four categories:

DB2 Guide and Reference Information
These books contain the common DB2 information for all platforms.

DB2 Installation and Configuration Information
These books are for DB2 on a specific platform. For example, there are separate *Quick Beginnings* books for DB2 on OS/2, Windows, and UNIX-based platforms.

Cross-platform sample programs in HTML
These samples are the HTML version of the sample programs that are installed with the Application Development Client. The samples are for informational purposes and do not replace the actual programs.

Release notes
These files contain late-breaking information that could not be included in the DB2 books.

The installation manuals, release notes, and tutorials are viewable in HTML directly from the product CD-ROM. Most books are available in HTML on the product CD-ROM for viewing and in Adobe Acrobat (PDF) format on the DB2 publications CD-ROM for viewing and printing. You can also order a printed copy from IBM; see "Ordering the Printed Books" on page 453. The following table lists books that can be ordered.

On OS/2 and Windows platforms, you can install the HTML files under the `sqllib\doc\html` directory. DB2 information is translated into different

languages; however, all the information is not translated into every language. Whenever information is not available in a specific language, the English information is provided

On UNIX platforms, you can install multiple language versions of the HTML files under the `doc/%L/html` directories, where *%L* represents the locale. For more information, refer to the appropriate *Quick Beginnings* book.

You can obtain DB2 books and access information in a variety of ways:

- "Viewing Information Online" on page 456
- "Searching Information Online" on page 460
- "Ordering the Printed Books" on page 453
- "Printing the PDF Books" on page 452

Table 53. DB2 Information

Name	Description	Form Number PDF File Name	HTML Directory
	DB2 Guide and Reference Information		
Administration Guide	*Administration Guide: Planning* provides an overview of database concepts, information about design issues (such as logical and physical database design), and a discussion of high availability.	SC09-2946 db2d1x70	db2d0
	Administration Guide: Implementation provides information on implementation issues such as implementing your design, accessing databases, auditing, backup and recovery.	SC09-2944 db2d2x70	
	Administration Guide: Performance provides information on database environment and application performance evaluation and tuning.	SC09-2945 db2d3x70	
	You can order the three volumes of the *Administration Guide* in the English language in North America using the form number SBOF-8934.		
Administrative API Reference	Describes the DB2 application programming interfaces (APIs) and data structures that you can use to manage your databases. This book also explains how to call APIs from your applications.	SC09-2947 db2b0x70	db2b0

Table 53. DB2 Information (continued)

Name	Description	Form Number PDF File Name	HTML Directory
Application Building Guide	Provides environment setup information and step-by-step instructions about how to compile, link, and run DB2 applications on Windows, OS/2, and UNIX-based platforms.	SC09-2948 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	Provides general information about APPC, CPI-C, and SNA sense codes that you may encounter when using DB2 Universal Database products. Available in HTML format only.	No form number db2apx70	db2ap
Application Development Guide	Explains how to develop applications that access DB2 databases using embedded SQL or Java (JDBC and SQLJ). Discussion topics include writing stored procedures, writing user-defined functions, creating user-defined types, using triggers, and developing applications in partitioned environments or with federated systems.	SC09-2949 db2a0x70	db2a0
CLI Guide and Reference	Explains how to develop applications that access DB2 databases using the DB2 Call Level Interface, a callable SQL interface that is compatible with the Microsoft ODBC specification.	SC09-2950 db2l0x70	db2l0
Command Reference	Explains how to use the Command Line Processor and describes the DB2 commands that you can use to manage your database.	SC09-2951 db2n0x70	db2n0
Connectivity Supplement	Provides setup and reference information on how to use DB2 for AS/400, DB2 for OS/390, DB2 for MVS, or DB2 for VM as DRDA application requesters with DB2 Universal Database servers. This book also details how to use DRDA application servers with DB2 Connect application requesters. Available in HTML and PDF only.	No form number db2h1x70	db2h1

Table 53. DB2 Information (continued)

Name	Description	Form Number PDF File Name	HTML Directory
Data Movement Utilities Guide and Reference	Explains how to use DB2 utilities, such as import, export, load, AutoLoader, and DPROP, that facilitate the movement of data.	SC09-2955 db2dmx70	db2dm
Data Warehouse Center Administration Guide	Provides information on how to build and maintain a data warehouse using the Data Warehouse Center.	SC26-9993 db2ddx70	db2dd
Data Warehouse Center Application Integration Guide	Provides information to help programmers integrate applications with the Data Warehouse Center and with the Information Catalog Manager.	SC26-9994 db2adx70	db2ad
DB2 Connect User's Guide	Provides concepts, programming, and general usage information for the DB2 Connect products.	SC09-2954 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	Provides an operational overview of the DB2 Query Patroller system, specific operational and administrative information, and task information for the administrative graphical user interface utilities.	SC09-2958 db2dwx70	db2dw
DB2 Query Patroller User's Guide	Describes how to use the tools and functions of the DB2 Query Patroller.	SC09-2960 db2wwx70	db2ww
Glossary	Provides definitions for terms used in DB2 and its components. Available in HTML format and in the *SQL Reference*.	No form number db2t0x70	db2t0
Image, Audio, and Video Extenders Administration and Programming	Provides general information about DB2 extenders, and information on the administration and configuration of the image, audio, and video (IAV) extenders and on programming using the IAV extenders. It includes reference information, diagnostic information (with messages), and samples.	SC26-9929 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	Provides guidance on managing information catalogs.	SC26-9995 db2dix70	db2di

Table 53. DB2 Information (continued)

Name	Description	Form Number PDF File Name	HTML Directory
Information Catalog Manager Programming Guide and Reference	Provides definitions for the architected interfaces for the Information Catalog Manager.	SC26-9997 db2bix70	db2bi
Information Catalog Manager User's Guide	Provides information on using the Information Catalog Manager user interface.	SC26-9996 db2aix70	db2ai
Installation and Configuration Supplement	Guides you through the planning, installation, and setup of platform-specific DB2 clients. This supplement also contains information on binding, setting up client and server communications, DB2 GUI tools, DRDA AS, distributed installation, the configuration of distributed requests, and accessing heterogeneous data sources.	GC09-2957 db2iyx70	db2iy
Message Reference	Lists messages and codes issued by DB2, the Information Catalog Manager, and the Data Warehouse Center, and describes the actions you should take. You can order both volumes of the Message Reference in the English language in North America with the form number SBOF-8932.	Volume 1 SC09-2978 db2m1x70 Volume 2 SC09-2979 db2m2x70	db2m0
OLAP Integration Server Administration Guide	Explains how to use the Administration Manager component of the OLAP Integration Server.	SC27-0782 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	Explains how to create and populate OLAP metaoutlines using the standard OLAP Metaoutline interface (not by using the Metaoutline Assistant).	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	Explains how to create OLAP models using the standard OLAP Model Interface (not by using the Model Assistant).	SC27-0783 db2lpx70	n/a
OLAP Setup and User's Guide	Provides configuration and setup information for the OLAP Starter Kit.	SC27-0702 db2ipx70	db2ip
OLAP Spreadsheet Add-in User's Guide for Excel	Describes how to use the Excel spreadsheet program to analyze OLAP data.	SC27-0786 db2epx70	db2ep

Table 53. DB2 Information (continued)

Name	Description	Form Number PDF File Name	HTML Directory
OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3	Describes how to use the Lotus 1-2-3 spreadsheet program to analyze OLAP data.	SC27-0785 db2tpx70	db2tp
Replication Guide and Reference	Provides planning, configuration, administration, and usage information for the IBM Replication tools supplied with DB2.	SC26-9920 db2e0x70	db2e0
Spatial Extender User's Guide and Reference	Provides information about installing, configuring, administering, programming, and troubleshooting the Spatial Extender. Also provides significant descriptions of spatial data concepts and provides reference information (messages and SQL) specific to the Spatial Extender.	SC27-0701 db2sbx70	db2sb
SQL Getting Started	Introduces SQL concepts and provides examples for many constructs and tasks.	SC09-2973 db2y0x70	db2y0
SQL Reference, Volume 1 and Volume 2	Describes SQL syntax, semantics, and the rules of the language. This book also includes information about release-to-release incompatibilities, product limits, and catalog views. You can order both volumes of the *SQL Reference* in the English language in North America with the form number SBOF-8933.	Volume 1 SC09-2974 db2s1x70 Volume 2 SC09-2975 db2s2x70	db2s0
System Monitor Guide and Reference	Describes how to collect different kinds of information about databases and the database manager. This book explains how to use the information to understand database activity, improve performance, and determine the cause of problems.	SC09-2956 db2f0x70	db2f0
Text Extender Administration and Programming	Provides general information about DB2 extenders and information on the administration and configuring of the text extender and on programming using the text extenders. It includes reference information, diagnostic information (with messages) and samples.	SC26-9930 desu9x70	desu9

Table 53. DB2 Information (continued)

Name	Description	Form Number PDF File Name	HTML Directory
Troubleshooting Guide	Helps you determine the source of errors, recover from problems, and use diagnostic tools in consultation with DB2 Customer Service.	GC09-2850 db2p0x70	db2p0
What's New	Describes the new features, functions, and enhancements in DB2 Universal Database, Version 7.	SC09-2976 db2q0x70	db2q0
DB2 Installation and Configuration Information			
DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings	Provides planning, migration, installation, and configuration information for DB2 Connect Enterprise Edition on the OS/2 and Windows 32-bit operating systems. This book also contains installation and setup information for many supported clients.	GC09-2953 db2c6x70	db2c6
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	Provides planning, migration, installation, configuration, and task information for DB2 Connect Enterprise Edition on UNIX-based platforms. This book also contains installation and setup information for many supported clients.	GC09-2952 db2cyx70	db2cy
DB2 Connect Personal Edition Quick Beginnings	Provides planning, migration, installation, configuration, and task information for DB2 Connect Personal Edition on the OS/2 and Windows 32-bit operating systems. This book also contains installation and setup information for all supported clients.	GC09-2967 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	Provides planning, installation, migration, and configuration information for DB2 Connect Personal Edition on all supported Linux distributions.	GC09-2962 db2c4x70	db2c4
DB2 Data Links Manager Quick Beginnings	Provides planning, installation, configuration, and task information for DB2 Data Links Manager for AIX and Windows 32-bit operating systems.	GC09-2966 db2z6x70	db2z6

Table 53. DB2 Information (continued)

Name	Description	Form Number PDF File Name	HTML Directory
DB2 Enterprise - Extended Edition for UNIX Quick Beginnings	Provides planning, installation, and configuration information for DB2 Enterprise - Extended Edition on UNIX-based platforms. This book also contains installation and setup information for many supported clients.	GC09-2964 db2v3x70	db2v3
DB2 Enterprise - Extended Edition for Windows Quick Beginnings	Provides planning, installation, and configuration information for DB2 Enterprise - Extended Edition for Windows 32-bit operating systems. This book also contains installation and setup information for many supported clients.	GC09-2963 db2v6x70	db2v6
DB2 for OS/2 Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database on the OS/2 operating system. This book also contains installation and setup information for many supported clients.	GC09-2968 db2i2x70	db2i2
DB2 for UNIX Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database on UNIX-based platforms. This book also contains installation and setup information for many supported clients.	GC09-2970 db2ixx70	db2ix
DB2 for Windows Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database on Windows 32-bit operating systems. This book also contains installation and setup information for many supported clients.	GC09-2971 db2i6x70	db2i6
DB2 Personal Edition Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database Personal Edition on the OS/2 and Windows 32-bit operating systems.	GC09-2969 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	Provides planning, installation, migration, and configuration information for DB2 Universal Database Personal Edition on all supported Linux distributions.	GC09-2972 db2i4x70	db2i4

Table 53. DB2 Information (continued)

Name	Description	Form Number PDF File Name	HTML Directory
DB2 Query Patroller Installation Guide	Provides installation information about DB2 Query Patroller.	GC09-2959 db2iwx70	db2iw
DB2 Warehouse Manager Installation Guide	Provides installation information for warehouse agents, warehouse transformers, and the Information Catalog Manager.	GC26-9998 db2idx70	db2id
Cross-Platform Sample Programs in HTML			
Sample programs in HTML	Provides the sample programs in HTML format for the programming languages on all platforms supported by DB2. The sample programs are provided for informational purposes only. Not all samples are available in all programming languages. The HTML samples are only available when the DB2 Application Development Client is installed. For more information on the programs, refer to the *Application Building Guide*.	No form number	db2hs
Release Notes			
DB2 Connect Release Notes	Provides late-breaking information that could not be included in the DB2 Connect books.	See note #2.	db2cr
DB2 Installation Notes	Provides late-breaking installation-specific information that could not be included in the DB2 books.	Available on product CD-ROM only.	
DB2 Release Notes	Provides late-breaking information about all DB2 products and features that could not be included in the DB2 books.	See note #2.	db2ir

Notes:

1. The character *x* in the sixth position of the file name indicates the language version of a book. For example, the file name `db2d0e70` identifies the English version of the *Administration Guide* and the file name `db2d0f70` identifies the French version of the same book. The following letters are used in the sixth position of the file name to indicate the language version:

Language	Identifier
Brazilian Portuguese	b

Bulgarian	u
Czech	x
Danish	d
Dutch	q
English	e
Finnish	y
French	f
German	g
Greek	a
Hungarian	h
Italian	i
Japanese	j
Korean	k
Norwegian	n
Polish	p
Portuguese	v
Russian	r
Simp. Chinese	c
Slovenian	l
Spanish	z
Swedish	s
Trad. Chinese	t
Turkish	m

2. Late breaking information that could not be included in the DB2 books is available in the Release Notes in HTML format and as an ASCII file. The HTML version is available from the Information Center and on the product CD-ROMs. To view the ASCII file:
   - On UNIX-based platforms, see the `Release.Notes` file. This file is located in the `DB2DIR/Readme/%L` directory, where `%L` represents the locale name and `DB2DIR` represents:
     - `/usr/lpp/db2_07_01` on AIX
     - `/opt/IBMdb2/V7.1` on HP-UX, PTX, Solaris, and Silicon Graphics IRIX
     - `/usr/IBMdb2/V7.1` on Linux.
   - On other platforms, see the `RELEASE.TXT` file. This file is located in the directory where the product is installed. On OS/2 platforms, you can also double-click the **IBM DB2** folder and then double-click the **Release Notes** icon.

Printing the PDF Books

If you prefer to have printed copies of the books, you can print the PDF files found on the DB2 publications CD-ROM. Using the Adobe Acrobat Reader, you can print either the entire book or a specific range of pages. For the file name of each book in the library, see Table 53 on page 444.

You can obtain the latest version of the Adobe Acrobat Reader from the Adobe Web site at http://www.adobe.com.

The PDF files are included on the DB2 publications CD-ROM with a file extension of PDF. To access the PDF files:

1. Insert the DB2 publications CD-ROM. On UNIX-based platforms, mount the DB2 publications CD-ROM. Refer to your *Quick Beginnings* book for the mounting procedures.
2. Start the Acrobat Reader.
3. Open the desired PDF file from one of the following locations:
   - On OS/2 and Windows platforms:

     x:\doc*language* directory, where *x* represents the CD-ROM drive and *language* represent the two-character country code that represents your language (for example, EN for English).
   - On UNIX-based platforms:

     /*cdrom*/doc/*%L* directory on the CD-ROM, where */cdrom* represents the mount point of the CD-ROM and *%L* represents the name of the desired locale.

You can also copy the PDF files from the CD-ROM to a local or network drive and read them from there.

Ordering the Printed Books

You can order the printed DB2 books either individually or as a set (in North America only) by using a sold bill of forms (SBOF) number. To order books, contact your IBM authorized dealer or marketing representative, or phone 1-800-879-2755 in the United States or 1-800-IBM-4YOU in Canada. You can also order the books from the Publications Web page at http://www.elink.ibmlink.ibm.com/pbl/pbl.

Two sets of books are available. SBOF-8935 provides reference and usage information for the DB2 Warehouse Manager. SBOF-8931 provides reference and usage information for all other DB2 Universal Database products and features. The contents of each SBOF are listed in the following table:

Table 54. Ordering the printed books

SBOF Number	Books Included	
SBOF-8931	• Administration Guide: Planning • Administration Guide: Implementation • Administration Guide: Performance • Administrative API Reference • Application Building Guide • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • Data Warehouse Center Administration Guide • Data Warehouse Center Application Integration Guide • DB2 Connect User's Guide • Installation and Configuration Supplement • Image, Audio, and Video Extenders Administration and Programming • Message Reference, Volumes 1 and 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volumes 1 and 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • What's New
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2 Online Documentation

Accessing Online Help

Online help is available with all DB2 components. The following table describes the various types of help.

Type of Help	Contents	How to Access...
Command Help	Explains the syntax of commands in the command line processor.	From the command line processor in interactive mode, enter: `? command` where *command* represents a keyword or the entire command. For example, `? catalog` displays help for all the CATALOG commands, while `? catalog database` displays help for the CATALOG DATABASE command.
Client Configuration Assistant Help *Command Center Help* *Control Center Help* *Data Warehouse Center Help* *Event Analyzer Help* *Information Catalog Manager Help* *Satellite Administration Center Help* *Script Center Help*	Explains the tasks you can perform in a window or notebook. The help includes overview and prerequisite information you need to know, and it describes how to use the window or notebook controls.	From a window or notebook, click the **Help** push button or press the **F1** key.

Message Help	Describes the cause of a message and any action you should take.	From the command line processor in interactive mode, enter:
		`? XXXnnnnn`
		where *XXXnnnnn* represents a valid message identifier.
		For example, `? SQL30081` displays help about the SQL30081 message.
		To view message help one screen at a time, enter:
		`? XXXnnnnn | more`
		To save message help in a file, enter:
		`? XXXnnnnn > filename.ext`
		where *filename.ext* represents the file where you want to save the message help.

Type of Help	Contents	How to Access...
SQL Help	Explains the syntax of SQL statements.	From the command line processor in interactive mode, enter: `help statement` where *statement* represents an SQL statement. For example, `help SELECT` displays help about the SELECT statement. **Note:** SQL help is not available on UNIX-based platforms.
SQLSTATE Help	Explains SQL states and class codes.	From the command line processor in interactive mode, enter: `? sqlstate or ? class code` where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state. For example, `? 08003` displays help for the 08003 SQL state, while `? 08` displays help for the 08 class code.

Viewing Information Online

The books included with this product are in Hypertext Markup Language (HTML) softcopy format. Softcopy format enables you to search or browse the information and provides hypertext links to related information. It also makes it easier to share the library across your site.

You can view the online books or sample programs with any browser that conforms to HTML Version 3.2 specifications.

To view online books or sample programs:

- If you are running DB2 administration tools, use the Information Center.
- From a browser, click **File —>Open Page**. The page you open contains descriptions of and links to DB2 information:
  - On UNIX-based platforms, open the following page:

    `INSTHOME/sqllib/doc/%L/html/index.htm`

    where *%L* represents the locale name.
  - On other platforms, open the following page:

    `sqllib\doc\html\index.htm`

    The path is located on the drive where DB2 is installed.

If you have not installed the Information Center, you can open the page by double-clicking the **DB2 Information** icon. Depending on the system you are using, the icon is in the main product folder or the Windows Start menu.

Installing the Netscape Browser

If you do not already have a Web browser installed, you can install Netscape from the Netscape CD-ROM found in the product boxes. For detailed instructions on how to install it, perform the following:

1. Insert the Netscape CD-ROM.
2. On UNIX-based platforms only, mount the CD-ROM. Refer to your *Quick Beginnings* book for the mounting procedures.
3. For installation instructions, refer to the `CDNAV`*nn*`.txt` file, where *nn* represents your two character language identifier. The file is located at the root directory of the CD-ROM.

Accessing Information with the Information Center

The Information Center provides quick access to DB2 product information. The Information Center is available on all platforms on which the DB2 administration tools are available.

You can open the Information Center by double-clicking the Information Center icon. Depending on the system you are using, the icon is in the Information folder in the main product folder or the Windows **Start** menu.

You can also access the Information Center by using the toolbar and the **Help** menu on the DB2 Windows platform.

The Information Center provides six types of information. Click the appropriate tab to look at the topics provided for that type.

Tasks Key tasks you can perform using DB2.

Reference DB2 reference information, such as keywords, commands, and APIs.

Books DB2 books.

Troubleshooting
Categories of error messages and their recovery actions.

Sample Programs
Sample programs that come with the DB2 Application Development Client. If you did not install the DB2 Application Development Client, this tab is not displayed.

Web DB2 information on the World Wide Web. To access this information, you must have a connection to the Web from your system.

When you select an item in one of the lists, the Information Center launches a viewer to display the information. The viewer might be the system help viewer, an editor, or a Web browser, depending on the kind of information you select.

The Information Center provides a find feature, so you can look for a specific topic without browsing the lists.

For a full text search, follow the hypertext link in the Information Center to the **Search DB2 Online Information** search form.

The HTML search server is usually started automatically. If a search in the HTML information does not work, you may have to start the search server using one of the following methods:

On Windows
Click **Start** and select **Programs —> IBM DB2 —> Information —> Start HTML Search Server**.

On OS/2
Double-click the **DB2 for OS/2** folder, and then double-click the **Start HTML Search Server** icon.

Refer to the release notes if you experience any other problems when searching the HTML information.

Note: The Search function is not available in the Linux, PTX, and Silicon Graphics IRIX environments.

Using DB2 Wizards

Wizards help you complete specific administration tasks by taking you through each task one step at a time. Wizards are available through the Control Center and the Client Configuration Assistant. The following table lists the wizards and describes their purpose.

Note: The Create Database, Create Index, Configure Multisite Update, and Performance Configuration wizards are available for the partitioned database environment.

Wizard	Helps You to...	How to Access...
Add Database	Catalog a database on a client workstation.	From the Client Configuration Assistant, click **Add**.
Back up Database	Determine, create, and schedule a backup plan.	From the Control Center, right-click the database you want to back up and select **Backup —> Database Using Wizard**.

Wizard	Helps You to...	How to Access...
Configure Multisite Update	Configure a multisite update, a distributed transaction, or a two-phase commit.	From the Control Center, right-click the **Databases** folder and select **Multisite Update**.
Create Database	Create a database, and perform some basic configuration tasks.	From the Control Center, right-click the **Databases** folder and select **Create —> Database Using Wizard**.
Create Table	Select basic data types, and create a primary key for the table.	From the Control Center, right-click the **Tables** icon and select **Create —> Table Using Wizard**.
Create Table Space	Create a new table space.	From the Control Center, right-click the **Table Spaces** icon and select **Create —> Table Space Using Wizard**.
Create Index	Advise which indexes to create and drop for all your queries.	From the Control Center, right-click the **Index** icon and select **Create —> Index Using Wizard**.
Performance Configuration	Tune the performance of a database by updating configuration parameters to match your business requirements.	From the Control Center, right-click the database you want to tune and select **Configure Performance Using Wizard**. For the partitioned database environment, from the Database Partitions view, right-click the first database partition you want to tune and select **Configure Performance Using Wizard**.
Restore Database	Recover a database after a failure. It helps you understand which backup to use, and which logs to replay.	From the Control Center, right-click the database you want to restore and select **Restore —> Database Using Wizard**.

Setting Up a Document Server

By default, the DB2 information is installed on your local system. This means that each person who needs access to the DB2 information must install the same files. To have the DB2 information stored in a single location, perform the following steps:

1. Copy all files and subdirectories from `\sqllib\doc\html` on your local system to a Web server. Each book has its own subdirectory that contains all the necessary HTML and GIF files that make up the book. Ensure that the directory structure remains the same.

2. Configure the Web server to look for the files in the new location. For information, refer to the NetQuestion Appendix in the *Installation and Configuration Supplement*.
3. If you are using the Java version of the Information Center, you can specify a base URL for all HTML files. You should use the URL for the list of books.
4. When you are able to view the book files, you can bookmark commonly viewed topics. You will probably want to bookmark the following pages:
   - List of books
   - Tables of contents of frequently used books
   - Frequently referenced articles, such as the ALTER TABLE topic
   - The Search form

For information about how you can serve the DB2 Universal Database online documentation files from a central machine, refer to the NetQuestion Appendix in the *Installation and Configuration Supplement*.

Searching Information Online

To find information in the HTML files, use one of the following methods:

- Click **Search** in the top frame. Use the search form to find a specific topic. This function is not available in the Linux, PTX, or Silicon Graphics IRIX environments.
- Click **Index** in the top frame. Use the index to find a specific topic in the book.
- Display the table of contents or index of the help or the HTML book, and then use the find function of the Web browser to find a specific topic in the book.
- Use the bookmark function of the Web browser to quickly return to a specific topic.
- Use the search function of the Information Center to find specific topics. See "Accessing Information with the Information Center" on page 457 for details.

Appendix M. Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

The following terms, which may be denoted by an asterisk(*), are trademarks of International Business Machines Corporation in the United States, other countries, or both.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

The following terms are trademarks or registered trademarks of other companies:

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

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

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, other countries, or both.

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

Other company, product, or service names, which may be denoted by a double asterisk(**) may be trademarks or service marks of others.

Index

Special Characters

$RAHBUFDIR 350
$RAHBUFNAME 350
$RAHCHECKBUF 350
$RAHENV 357

A

access control 225
  authentication 225
  database manager 256
  database objects 255
  view to table 261
active directory 395
  configuring 397
  extending the directory schema 411
  objects 413
  security 409
  supporting 396
add database wizard 458, 459
adding a scope 189
adding constraint 191
adding table check constraint 193
adding unique constraint 192
administration
  using GUI tools 3
administration server 14, 78
administration tools
  command center 19
  overview 4
  script center 20
aggregating function 139
alias
  authority 150
  using 149
alias, creating 149
alias (DB2 for MVS/ESA) 151
ALTER COLUMN 189
ALTER NICKNAME statement, example of 213
ALTER privilege
  definition 251
ALTER SERVER statement, example of 212
alter summary table properties 203
ALTER TABLE statement
  adding check constraint example 194
  adding columns example 189
ALTER TABLE statement *(continued)*
  adding keys example 193
  adding unique constraint example 192
  dropping check constraint example 196
  dropping keys example 195
  dropping unique constraint example 194
  tips for adding constraints 192
ALTER TABLESPACE statement
  example of 183
ALTER VIEW statement; example of 209
altering a column 189
altering a nickname 213
altering a server 212
altering a structured type 204
altering a table 188
altering an IDENTITY column 191
altering constraint 191
altering nodegroup 182
altering table space 182
altering view 209
ATTACH command
  overview of 54
  specifying distributed computing environment (DCE) information 332
audit activities 273
audit_buf_sz 275
audit facility
  actions 274
  asynchronous record writing 276
  audit events table 282
  authorities/privileges 273
  behavior 275
  checking events table 283
  CONTEXT events table 296
  controlling activities 299
  error handling 276
  ERRORTYPE parameter 276
  events 274
  examples 299
  messages 281
  OBJMAINT events table 287
  parameter descriptions 278
  record layouts 282
audit facility *(continued)*
  SECMAINT events table 288
  synchronous record writing 276
  syntax 277
  SYSADMIN events table 292
  tips and techniques 297
  usage scenarios 277
  VALIDATE events table 295
audit trail 273
authentication 225
  DCE security services 231
  definition of 225
  Distributed Computing Environment (DCE) directory services 329
  domain security 366
  federated database processing 237
  groups 366
  partitioned database considerations 230
  remote client 230
authentication type 225
  CLIENT 226
  DCE 228
  DCE_SERVER_ENCRYPT 228
  DCS 227
  DCS_ENCRYPT 228
  KERBEROS 229
  KRB_SERVER_ENCRYPT 229
  SERVER 225
  SERVER_ENCRYPT 225
authorities
  granting 38
  revoking 38
authority 244
  database administration (DBADM) 247, 249
  levels of 242
  removing DBADM from SYSADM 245
  removing DBADM from SYSCTRL 246
  system control (SYSCTRL) 245
  system maintenance (SYSMAINT) 246
  tasks and required authorities 266

authorization
definition 242
system administration (SYSADM) 244
trusted client 226
authorization names
create view for privileges information 270
retrieving for privileges information 268
retrieving names with DBADM authority 268
retrieving names with table access authority 268
retrieving privileges granted to 269
automatic summary table 147

B

back up database wizard 458
backup database wizard 5
backup domain controller
configuring DB2 364, 365
BIND command
OWNER option 259
BIND privilege
definition of 255
BINDADD privilege
definition 248
binding
command line processor 109
database utilities 109
rebinding invalid packages 259
block-structured devices 112
books 443, 453

C

Call Level Interface (CLI)
binding to a database 109
case-sensitive names, federated database 317
CATALOG DATABASE
example of 110
CATALOG GLOBAL DATABASE
specifying distributed computing environment (DCE) information 332
catalog node
description 57
cataloging database 109
CDS 319
cell directory service (CDS) 319
changing database configuration 179
changing environment variables 179
changing node configuration file 179
changing partitioning key 200
changing registry variables 179
changing table attributes 201
character serial devices 112
CLIENT, authentication type 226
CLIENT level security 226
clients
administration 4
trusted 226
untrusted 226
collating_sequence server option 155
column options
numeric string 214
varchar_no_trailing_blanks 214
column UDF 139
columns
adding 188
altering 189
defining 119
comm_rate server option 155
command center 19
command line processor
binding to a database 109
commands
CATALOG GLOBAL DATABASE 332
running in parallel 350
communication protocols
VI architecture 385
communications
high speed 383
configuration parameter
partitioned database 58
configuration parameters
distributed computing environment (DCE) 331
configure multisite update wizard 5, 458
configuring LDAP 397
configuring LDAP user for applications 399
CONNECT privilege
definition 248
CONNECT statement
specifying distributed computing environment (DCE) information 332
connectstring server option 156
constraint
adding 191
constraint *(continued)*
changing 191
defining unique 123
dropping 194
constraint name
defining foreign keys 126
defining table check constraints 127
containers
adding (to DMS table space) 182
adding to an SMS table space 185
modifying (to DMS table space) 183
control center
customized 12
displaying systems 14
control center as a java applet 46
CONTROL privilege
definition 251
implicit issuance 259
package privileges 255
controlling the rah command 356
cooked devices 112
cpu_ratio server option 156
CREATE ALIAS statement
example of 150
using 149
CREATE DATABASE command
example of 101
create database wizard 5, 459
create index 161
CREATE INDEX statement
example of 165
online reorganization 164, 167
unique index 166
CREATE NICKNAME 159
CREATE_NOT_FENCED privilege
definition 248
CREATE SERVER 152
create table space wizard 5, 459
CREATE TABLE statement
defining check constraints 127
defining referential constraints 124
example of 120
using multiple table spaces 134
create table wizard 5, 459
CREATE TABLESPACE statement
example of 111
CREATE TRIGGER statement
example of 137
CREATE VIEW statement
changing column names 146

CREATE VIEW statement *(continued)*
example of 145
CREATE WRAPPER 151
CREATETAB privilege
definition 248
creating a function mapping 140
creating a function template 140
creating a nickname 159
creating a server 152
creating a type mapping 143
creating a wrapper 151
creating alias 149
creating an index extension 161
creating an index specification 161
creating an LDAP user 398
creating schema 116
creating table 118
creating table in multiple table spaces 134
creating table space 111
creating trigger 136
creating typed table 133
creating typed view 147
creating user-defined distinct type 142
creating user-defined function 138
creating user-defined structured type 143
creating user-defined type 142
creating view 144
CURRENT SCHEMA 118
CURRENT SCHEMA special register 55

D

data
changing distribution 182
moving 39
data definition language (DDL)
generating 8
data encryption 264
data integrity
unique index 161
data replication 305
data security
controlling database access 221
importance of 221
securing system catalog 270
data transfer
overview of 305
data types
column definition 119
multi-byte character set 119
database 51
altering nodegroup 182
database *(continued)*
before creating 51
cataloging 109
changing 181
changing distribution of data 182
considerations before changing 175
considerations for creating 60
creating 101
creating across all nodes 58
dropping 181
enabling data partitioning 57
package dependencies 216
recovery log 109
database access
controlling 221
privileges through package with SQL 260
database administrator (DBADM) authority
privileges 247
retrieving names with 268
database configuration
changing 179
created file 97
database locator objects
creating 321
example 322
database manager
access control 256
binding utilities 109
index 164
starting 52
stopping 59
database objects
access control 255
creating 320
example 320
database partition servers
issuing commands 347
Windows 2000 375
Windows NT 375
databases
remote, managing 36
DataPropagator Relational (DPROPR)
overview 305
DAU (DB_Authentication) 325
DB_Authentication (DAU) 325
DB_Comment (DCO) 325
DB_Communication_Protocol (DCP) 325
DB_Database_Locator_Name (DLN) 327
DB_Database_Protocol (DDP) 327
DB_Native_Database_Name (DNN) 327
DB_Object_Type (DOT) 327
DB_Principal (DPR) 325
DB_Product_Name (DPN) 327
DB_Product_Release (DRL) 327
DB_Target_Database_Info (DTI) 327
DB2
starting on Windows NT 53
DB2 Administration Server
update configuration 94
update instance lists 94
using Client Configuration Assistant and Control Center 93
DB2 Administration Server (DAS) 84
communications 87
configuration 86
configuring 82
Control Center
communications 87
creating 79
enabling discovery of 90
environment 89
internode administrative communications 87
internode administrative communications in partitioned database system (UNIX) 87
internode administrative communications in partitioned database system (Windows NT) 89
listing 82
overview 78
ownership rules 77
registry variable considerations 89
registry variables 89
removing 83
security 89
security considerations 82
service ports 87
setting up with partitioned database system 84
example 85
starting and stopping 81
UNIX EEE servers 87
updating 83
Windows NT EEE servers 89
db2_all 347, 348, 349
db2_call_stack 348
DB2 Connect 305

DB2 for OS/390
objects, managing 15
subsystems, adding 15
DB2 for Windows NT Performance Counters 369
DB2_INDEX_2BYTEVARLEN 162
db2_kill 348
DB2 library
books 443
Information Center 457
language identifier for books 451
late-breaking information 452
online help 454
ordering printed books 453
printing PDF books 452
searching online information 460
setting up document server 459
structure of 443
viewing online information 456
wizards 458
db2audit 277
db2audit.log 273
db2dmnbckctlr
using 364, 365
db2gncol utility 197
db2icrt command 65
db2idrop 178
db2ilist 176
DB2INSTANCE environment variable
defining default instance 54
db2iupdt 176
DB2LDAP_CLIENT_PROVIDER 396
db2ldcfg utility 399
db2nchg 377
db2ncrt 375
db2ndrop 378
db2nlist 375
db2nodes.cfg file 95
db2perfc 372
db2perfi 369
db2perfr 370
db2set command 70, 71
db2start command 52
db2stop command 59
dbname server option 156
DCE, authentication type 228
DCE network database
connecting 335, 336
creating 334
DCE_SERVER_ENCRYPT, authentication type 228
DCO (DB_Comment) 325
DCP (DB_Communication_Protocol) 325
DCS
authentication type 227
federated database processing 237
DCS_ENCRYPT, authentication type 228
DDP (DB_Database_Protocol) 327
DECLARE GLOBAL TEMPORARY TABLE 129
dedicated interconnect 385, 386
default attribute specification 120
defining referential constraint 124
defining table check constraint 127
defining unique constraint 123
DELETE privilege
definition 251
deleting rows from typed tables 204
design, implementing 51
design of database
altering 175
DETACH command
overview of 54
determining problems with rah 359
directories
local database directory 105
node directory 106
system database directory 106
directory cache
effect of cataloging databases 110
directory objects
creating 319
object classes attributes 324
Discovery
configuration 94
hiding server instances 92
setting parameters 92
distributed computing environment (DCE)
authentication 231
security services 231
setup DB2 server 233
setup DB2 user 231
Distributed Computing Environment (DCE)
ATTACH command 332, 337
CATALOG GLOBAL DATABASE 332
CDS 319
configuration parameters and registry variables 331
CONNECT statement 332, 338
Distributed Computing Environment (DCE) *(continued)*
directory services restrictions 343
directory services tasks 340
GDS 319
how directories are searched 337
overview of directory services 107
restrictions 235
setup DB2 client instance 235
temporarily overriding DCE directory information 339
using directory services 341
DLN (DB_Database_Locator_Name) 327
DMS table space
creating 112
DNN (DB_Native_Database_Name) 327
domain security
authentication 366
DOT (DB_Object_Type) 327
double byte character set user data type 119
DPN (DB_Product_Name) 327
DPR (DB_Principal) 325
DPROPR 305
DRL (DB_Product_Release) 327
drop a system temporary table space 186
drop a user temporary table space 187
DROP DATABASE command
example of 181
DROP INDEX statement; example of 215
DROP NICKNAME statement, example of 213
DROP SERVER statement, example of 212
DROP TABLE statement
example of 205
DROP TABLESPACE statement; example of 186
DROP VIEW statement; example of 209
dropping a nickname 213
dropping a server 212
dropping a summary table 210
dropping a wrapper 211
dropping constraint 194
dropping database 181
dropping index 214

dropping index extensions 214
dropping index specifications 214
dropping schema 188
dropping table 205
dropping table check constraint 196
dropping trigger 207
dropping unique constraint 194
dropping user-defined function 207
dropping user-defined tables 207
dropping user-defined type 208
dropping user table space 186
dropping view 209
DTI (DB_Target_Database_Info) 327
dynamic SQL
  EXECUTE privilege for database access 260

E

eliminating duplicate entries from machine list 355
encryption
  data 264
environment variables 70
  changing 179
  rah 356
  RAHDOTFILES 357
  setting on OS/2 73
  setting on UNIX 76
  setting on Windows 95 74
  setting on Windows NT 74
EXECUTE privilege
  database access with dynamic SQL 260
  database access with static SQL 260
  definition of 255
explicit schema use 55
expressions
  NEXTVAL 132
  PREVVAL 132

F

FCM communications 98
federated database
  case-sensitive names 317
  function mapping, creating 140
  function template, creating 140
  index specification, creating 161
  nickname, creating 159
  nickname, identifying 160
  nickname, working with 160
  passing IDs and passwords to data sources 238
  referencing nicknames 160
  server, creating 152
federated database *(continued)*
  type mapping, creating 143
  wrapper, creating 151
federated databases
  APPC setting 240
  authentication 237
  authentication example 240
  DCS setting 237
  server options, security 239
  user mappings, creating 238
filters 9
fold_id server option 156
fold_pw server option 156
FOREIGN KEY clause
  referential constraints 125
  rules for foreign key definitions 125
foreign keys
  adding 192
  composite 125
  constraint name 125
  DROP FOREIGN KEY clause, ALTER TABLE statement 194
  IMPORT utility, referential integrity implications for 127
  load utility, referential integrity implications for 126
  privileges required for dropping 194
  rules for foreign key definitions 125
function invocation
  selectivity 173
function mapping, creating 140
function template, creating 140
functions
  DECRYPT 265
  ENCRYPT 264
  GETHINT 265

G

gateway connections 16
GDS 319
generated column 127, 196
global directory service (GDS) 319
global level profile registry 70
GRANT
  example 256
GRANT statement
  implicit issuance 259
  security 331
  use of 256
granting authorities and privileges 38
GUI tools
  administring using 3

H

hierarchy table 134
  dropping 206
high speed communications 383
HTML
  sample programs 451

I

IBM eNetwork Directory
  extending the directory schema 411
  object classes and attributes 413
IBMCATGROUP nodegroup 103
IBMDEFAULTGROUP nodegroup 103
IBMTEMPGROUP nodegroup 103
identifying nicknames 160
identity column 130
  altering 201
IDENTITY columns 133
IMPLICIT_SCHEMA authority 117
IMPLICIT_SCHEMA privilege
  definition 248
implicit schema use 55
IMPORT utility
  binding to a database 109
  LOAD 126
  referential integrity implications for 127
index
  selectivity 173
  user-defined extended 168
index extension 161
index key, definition 162
INDEX privilege
  definition 251
index wizard 5, 459
indexes
  changing 214
  CREATE INDEX statement 165
  CREATE UNIQUE INDEX statement 166
  creating 161
  definition of 162
  DROP INDEX statement 215
  how used 165
  non-unique 165
  nonprimary 215
  online reorganization 164, 167
  optimizing number 162
  primary 123
  primary versus user-defined 162

indexes *(continued)*
privileges 255
unique 166
Information Center 457
INSERT privilege
definition 251
installing
Netscape browser 457
instance
add 65
default 62
definition 61
directory 61
owner 64
removing 178
setting the current 68
instance level profile registry 70
instance owner 64
instance profile registry 71
instance user
setting the environment 62
instances
altering 175
auto-starting 69
creating 62
disadvantages 61
displaying 15
listing 68, 176
listing database partition servers 375
overview of 53
partition servers, adding 375
partition servers, changing 377
partition servers, dropping 378
reasons for using 61
running multiple 69
starting 52
stopping 59
updating 176
inter-node communications 383
intra-partition parallelism
enabling 55
io_ratio server option 157

J

java applet 46

K

KERBEROS, authentication type 229
Kerberos security protocol 229
KRB_SERVER_ENCRYPT, authentication type 229

L

language identifier
books 451
Large Object (LOB)
column considerations 121
late-breaking information 452
LDAP 45, 107, 395
LDAP configurations
supporting 395
License Center 22
license information
altering 175
license management 70
lightweight directory access protocol 107
attaching remotely 402
cataloging a node entry 401
configure host databases 405
DB2 Connect 408
deregistering databases 403
deregistering servers 402
disable 408
enable 407
extending directory schema 410
IBM eNetwork Directory 411
object classes and attributes 413
refreshing entries 403
registering databases 402
searching 405
security 408
setting registry variables 407
updating protocol information 401
Windows 2000 active directory 411
lightweight directory access protocol (LDAP) 45, 395
LOAD authority 247
LOAD utility
overview 305
local database directory
overview of 105
locate objects 18
log
audit 273
logging
raw devices 114
logical nodes
multiple 381

M

messages
audit facility 281
MINPCTUSED clause 167
modifying a column 189
modifying a table 188
monitoring
rah processes 351
moving data 305
multiple instances 53

N

naming conventions
general 313
Windows NT restrictions 364
Netscape browser
installing 457
NEXTVAL 132
nickname
creating 159
nicknames
package privilege processing 260
privileges 253
views across data sources 261
node 55
cataloging 57
changing in nodegroup 182
creating database across all 58
node configuration file
changing 179
creating 95
node level profile registry 71
node number 95
node server option 157
nodegroup
initial definition 103
nodegroups
altering 182
creating 108
IBMDEFAULTGROUP, table created in by default 135
partitioning key, changing 200
table considerations 135
non-unique indexes
dropping 215
nonprimary indexes
dropping 215
dropping implications for applications 215
null value
column definition 119
numeric string column option 214

O

object class attributes
DB_Authentication (DAU) 325
DB_Comment (DCO) 325
DB_Communication_Protocol 325
DB_Database_Locator_Name 327

object class attributes *(continued)*
DB_Database_Protocol 327
DB_Native_Database_Name 327
DB_Object_Type 327
DB_Principal (DPR) 325
DB_Product_Name 327
DB_Product_Release 327
DB_Target_Database_Info 327
objects
show related 7
online help 454
online information
searching 460
viewing 456
online reorganization
indexes 164
opening the Journal 22

P

package
inoperative 216
packages
access privileges with SQL 260
dependencies 215
dropping 215
invalid after adding foreign key 192
owner 259
privileges 254
revoking privileges 258
parallelism
enabling 55
parallelism, intra-partition
enabling 55
partitioning data 57
partitioning key
changing 200
index partitioned on partitioning key 164
table considerations 135
passing IDs and passwords to data sources 238
password server option 157
PDF 452
performance
accessing remote information 372
catalog information, reducing contention for 58
displaying information 370
enable remote access to information 370
resetting values 372
summary table 147
performance configuration
wizard 5, 459
Performance Configuration Wizard 179
performance monitor
Windows NT 369
plan_hints server option 157
populating typed table 133
PRECOMPILE command
OWNER option 259
prefix sequences 355
PREVVAL 132
primary index
dropping 215
uniqueness for primary key 123
primary key
adding 192
DROP PRIMARY KEY clause, ALTER TABLE statement 194
primary index 123
primary index, creating 162
privileges required for dropping 194
when to create 123
PRIMARY KEY clause
adding primary key 192
restrictions 123
printing PDF books 452
privileges
ALTER 251
BINDADD 248
CONNECT 248
CONTROL 251
CREATE_NOT_FENCED 248
create view for information 270
CREATETAB 248
database manager 248
definition of 242
DELETE 251
GRANT statement 256
granting 38
granting and revoking authority 248
hierarchy 243
implicit for packages 244
IMPLICIT_SCHEMA 248
INDEX 255
indirect privileges, nicknames 260
individual 244
INSERT 251
nickname 253
ownership (CONTROL) 244
package 254
PUBLIC 249
privileges *(continued)*
REFERENCES 251
retrieving authorization names with 268
retrieving for names 269
REVOKE statement 257
revoking 38
schemas 249
SELECT 251
servers 254
summary of 243
system catalog listing 267
table 251
table spaces 250
tasks and required authorities 266
USAGE 255
view 251
views with nicknames 261
profile registry 70
PUBLIC
privileges 249
public interconnect 384
pushdown server option 158

Q

qualified object names 54
query rewrite
summary table 147

R

rah 348
controlling 356
environment variables 356
introduction 347
monitoring processes 351
RAHDOTFILES 357
RAHOSTFILE 355
RAHOSTLIST 355
RAHWAITTIME 351
running commands in parallel 350
setting default environment profile 358
specifying machines list 355
RAHCHECKBUF 351
RAHTREETHRESH 352
raw devices 112
raw I/O 114
raw logs 114
records
audit 273
recovering inoperative summary table 211
recovering inoperative view 210

recovery
allocating log during database creation 109
overview 309, 345
recovery log 109
redistributing data
across nodes 182
REFERENCES clause
adding foreign key 192
delete rules 126
referential constraints 126
use of 126
REFERENCES privilege
definition 251
referential constraints
add to table 192
defining 124
FOREIGN KEY clause, CREATE/ALTER TABLE statements 124
PRIMARY KEY clause, CREATE/ALTER TABLE statements 124
REFERENCES clause, CREATE/ALTER TABLE statements 124
refreshing data in summary table 204
registry variables 70
changing 179
distributed computing environment (DCE) 331
release notes 452
remote administration 84
remote system 37
renaming a table space 185
renaming table 204
REORG utility
binding to a database 109
replicating data 44
replication 305
configuration 98
resource access control facility (RACF) 331
restore database wizard 5
restore wizard 459
restrictions
Windows NT naming 364
retrieving data
index 164
REVOKE statement
example of 257
implicit issuance 259
security 331
use of 257
revoking authorities and privileges 38
routing information objects
creating 322
example 323
rows
removing 190

S

sample programs
cross-platform 451
HTML 451
scalar UDF 138
scheduling
saved command scripts 21
schema
creating 116
dropping 188
overview of 54
SESSION 207
scope
adding 189
script center 20
using an existing script 21
Search Discovery
additional settings 91
searching
online information 458, 460
security
APPC setting for federated systems 240
authentication, federated database 237
CLIENT level 226
DCS processing, federated system 237
Distributed Computing Environment (DCE) directory services 329
federated database ID and password processing 238
federated server authentication example 240
planning for 221
server options 239
services, Windows NT 365
user mappings 238
SELECT privilege
definition 251
SELECT statement
select a view 146
selectivity 173
sequences 133
altering 202
creating 131
sequences *(continued)*
dropping 202
privileges 255
server
creating 152
SERVER, authentication type 225
SERVER_ENCRYPT
authentication type 225
server options
collating_sequence 155
comm_rate 155
connectstring 156
cpu_ratio 156
dbname 156
fold_id 156, 240
fold_pw 156, 240
io_ratio 157
node 157
password 157, 239
plan_hints 157
pushdown 158
security details 239
varchar_no_trailing_blanks 158
servers
privileges 254
SET ENCRYPTION PASSWORD 265
setting
default environment profile for rah 358
setting schema 118
setting up document server 459
setting VARCHAR 189
show related objects 7
show SQL statements 7
SIGTTIN 349
size
estimating 41
SmartGuides
wizards 458
SMS table space
creating 111
SMS table spaces
adding containers 185
sparse file allocation 122
SQL statements
inoperative 216
show 7
starting DB2 52
static SQL
EXECUTE privilege for database access 260
stdin 349
stopping DB2 59

storage
management 41
structured type
altering 204
summary table
altering properties 203
automatic 147
creating 147
dropping 210
refreshing data 204
summary tables
recovering inoperative 211
SWITCH ONLINE clause 185
synonym (DB2 for MVS/ESA) 151
SYSCAT views 267
SYSCATSPACE table space 103
system administration (SYSADM)
authority 244
overview 244
privileges 244
system catalog
adding new column 189
dropping a table 205
dropping view implications 210
privileges listing 267
retrieving authorization names with privileges 268
retrieving names with DBADM authority 268
retrieving names with table access authority 268
retrieving privileges granted to names 269
security 270
setting up 104
system catalog tables
stored on database catalog node 58
system database directory
overview of 106
system temporary table space 113

T

table
altering 188
changing attributes 201
creating in partitioned database 135
generated column 127, 196
identity column 130
renaming 204
temporary 103
volatile 199
table check constraint
adding 193
table check constraint *(continued)*
defining 127
dropping 196
table space
adding container 182
changing 182
creating 111
default at database creation 103
device container example 112
dropping 186
dropping a system temporary 186
dropping a user temporary 187
extending container 183
file container example 111
file system container example 111
in nodegroups 114
privileges 250
renaming 185
resizing container 183
separating types of data, example 134
system temporary 113
user temporary 114
table spaces
adding capacity 43
checking available space (DMS) 42
table UDF 139
tables
add referential constraints 192
ALTER TABLE statement 189
assigning to nodegroup 108
changing partitioning key 200
CREATE TABLE statement 118
defining check constraint 127
defining referential constraints 124
defining unique constraint 123
dropping 205
naming 118
retrieving names with access to 268
revoking privileges 257
tablespaces
setting to ONLINE state 185
TCP/IP 384
temporary table
user-defined 129
temporary tables
dropping user-defined 207
TEMPSPACE1 table space 103
trail
audit 273
trigger
dropping 207
triggers
benefits of 136
creating 136
dependencies 138
troubleshooting 43
trusted clients
authentication 226
CLIENT level security 226
type mapping, creating 143
typed table
creating 133
hierarchy table 134
populating 133
updating rows 204
typed tables
deleting rows 204
typed view, creating 147

U

unique constraints
adding 192
defining 123
dropping 194
untrusted clients 226
update DAS configuration 94
update instance lists 94
UPDATE privilege
definition 252
updating typed table 204
USAGE privilege 255
user authentication
Windows NT 364
user-defined distinct type, creating 142
user-defined functions (UDF)
creating 138
dropping 207
types 138
user-defined functions (UDFs)
privilege to create non-fenced 248
user-defined structured type, creating 143
user-defined temporary table 129
user-defined temporary tables 207
user-defined type (UDT)
creating 142
dropping 208
user mappings
creating 238
user temporary table space 114
users
managing 38

USERSPACE1 table space 103

V

varchar_no_trailing_blanks column option 214
varchar_no_trailing_blanks server option 158
VI
  setting up for use with DB2 393
VI architecture 385
view
  recovering inoperative 210
viewing
  online information 456
views
  access control to table 261
  access privileges, examples of 262
  altering 209
  CHECK OPTION clause, CREATE VIEW statement 146
  column access 261
  creating 144
  data integrity 144
  data security 144
  dropping 209
  dropping implications for system catalogs 209
  for privileges information 270
  inoperative 210
  restrictions 209
  row access 261
virtual interface (VI)
  architecture 385, 386
virtual telecommunications access method (VTAM) 331

W

Windows 2000 active directory
  extending the directory schema 411
  objects 413
Windows NT active directory
  object classes and attributes 413
Windows NT performance monitor 369
  registering DB2 369
wizard
  restore database 459
Wizard
  Performance Configuration 179
wizards 5
  add database 458, 459
  back up database 458
  backup database 5
wizards *(continued)*
  completing tasks 458
  configure multisite update 5, 458
  create database 5, 459
  create table 5, 459
  create table space 5, 459
  index 5, 459
  performance configuration 5, 459
  restore database 5
wrapper, creating 151

Contacting IBM

If you have a technical problem, please review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. This guide suggests information that you can gather to help DB2 Customer Support to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

Product Information

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

http://www.ibm.com/software/data/
The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more.

http://www.ibm.com/software/data/db2/library/
The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information.

Note: This information may be in English only.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
The International Publications ordering Web site provides information on how to order books.

http://www.ibm.com/education/certify/
The Professional Certification Program from the IBM Web site provides certification test information for a variety of IBM products, including DB2.

ftp.software.ibm.com
Log on as anonymous. In the directory `/ps/products/db2`, you can find demos, fixes, information, and tools relating to DB2 and many other products.

comp.databases.ibm-db2, bit.listserv.db2-l
These Internet newsgroups are available for users to discuss their experiences with DB2 products.

On Compuserve: GO IBMDB2
Enter this command to access the IBM DB2 Family forums. All DB2 products are supported through these forums.

For information on how to contact IBM outside of the United States, refer to Appendix A of the *IBM Software Support Handbook*. To access this document, go to the following Web page: http://www.ibm.com/support/, and then select the IBM Software Support Handbook link near the bottom of the page.

Note: In some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC09-2944-01