IBM® DB2 Universal Database™

Administration Guide: Implementation

Version 8.2

SC09-4820-01

IBM® DB2 Universal Database™

Administration Guide: Implementation

Version 8.2

SC09-4820-01

Before using this information and the product it supports, be sure to read the general information under *Notices*.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

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

Contents

About this book

The Administration Guide in its three volumes provides information necessary to use and administer the 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 use a graphical user interface to 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, which allows you 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, select the Control Center icon from the DB2 folder, or use the Start menu on Windows platforms. For introductory help, select **Getting started** from the **Help** pull-down of the Control Center window. The **Visual Explain** tool is invoked from the Control Center.

  The Control Center is available in three views:

  – Basic. This view shows the core DB2 UDB functions on essential objects such as databases, tables, and stored procedures.
  – Advanced. This view has all of the objects and actions available. Use this view if you are working in an enterprise environment and you want to connect to DB2 for z/OS or IMS.
  – Custom. This view gives you the ability to tailor the object tree and the object actions.

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

- The Command Editor which replaces the Command Center and is used to generate, edit, run, and manipulate SQL statements; IMS and DB2 commands; work with the resulting output; and to view a graphical representation of the access plan for explained SQL statements.

- The Development Center to provide support for native SQL Persistent Storage Module (PSM) stored procedures; for Java stored procedures for iSeries Version 5 Release 3 and later; user-defined functions (UDFs); and structured types.
- The Health Center provides a tool to assist DBAs in the resolution of performance and resource allocation problems.
- The Tools Settings to change the settings for the Control Center, Health Center, and Replication Center.
- 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 Universal Database™ (DB2 UDB) relational database management system.

How this book is structured

This book contains information about the following major topics:

Implementing Your Design

- Chapter 1, "Before creating a database," describes the prerequisites needed before creating a database and the objects within a database.
- Chapter 2, "Creating and using the DB2 Administration Server (DAS)," discusses what a DAS is, how to create it, and how to use it.
- Chapter 3, "Creating a database," describes the tasks associated with creating a database and the objects within a database.
- Chapter 4, "Creating tables and other related table objects," describes how to create tables with specific characteristics when implementing your database design.
- Chapter 5, "Altering a database," describes the prerequisites and the tasks associated with altering or dropping a database and the objects within a database.
- Chapter 6, "Altering tables and other related table objects," describes how to drop tables or how to modify specific characteristics associated with those tables. Dropping and modifying related table objects is also presented here.

Database Security

- Chapter 7, "Controlling database access," describes how you can control access to your database's resources.
- Chapter 8, "Auditing DB2 Universal Database™ (DB2 UDB) activities," describes how you can detect and monitor unwanted or unanticipated access to data.

Appendixes

- Appendix A, "Conforming to the naming rules," presents the rules to follow when naming databases and objects.
- Appendix B, "Using automatic client rerouting," discusses the automatic rerouting of client applications and how to enable this support.

- Appendix C, "Using lightweight directory access protocol (LDAP) directory services," provides information about how you can use LDAP Directory Services.
- 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, "Using Windows Management Instrumentation (WMI) support," provides information about how DB2 can be managed using WMI.
- Appendix F, "Using Windows NT security," describes how DB2 works with Windows security.
- Appendix G, "Using the Windows Performance Monitor," describes how to use the Windows Performance Monitor to collect DB2 performance data.
- Appendix H, "Using Windows database partition servers," describes the utilities used by Windows to work with partitioned database servers.
- Appendix I, "Configuring multiple logical nodes," describes how to configure multiple logical nodes in a partitioned database environment.
- Appendix J, "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.

A chapter was moved from the *Administration Guide: Implementation* manual that had the title "Utilities for Moving Data".

Note: All of the information on the DB2 utilities for moving data, 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 of data, see *IBM DB2 Information Integrator SQL Replication Guide and Reference*.

A chapter was moved from the *Administration Guide: Implementation* manual that had the title "Recovering a Database".

Note: All of the information on the methods and tools for backing up and recovering data, 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.

A brief overview of the other Administration Guide volumes

Administration Guide: Planning

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

Database Concepts

- "Basic relational database concepts" presents an overview of database objects, including recovery objects, storage objects, and system objects.
- "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.

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.
- "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.

Appendixes

- "Incompatibilities between releases" presents the incompatibilities introduced by Version 7 and Version 8, as well as future incompatibilities that you should be aware of.
- "National language support (NLS)" describes DB2 National Language Support, including information about territories, languages, and code pages.
- "Enabling large page support in a 64-bit environment (AIX)" discusses the support for a 16 MB page size and how to enable this support.

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

- "Introduction to Performance" introduces concepts and considerations for managing and improving DB2 UDB performance.
- "Architecture and Processes" 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 database manager, database, and DAS 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. Implementing Your Design

Chapter 1. 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, database partition groups, table spaces, tables, views, wrappers, servers, nicknames, type mappings, function mappings, aliases, user-defined types (UDTs), user-defined functions (UDFs), automatic summary tables (ASTs), triggers, constraints, indexes, and packages. You can create these objects using SQL statements in the command line processor and through SQL statements 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* manual. For information on application programming interfaces (APIs), refer to the *Administrative API Reference* manual.

Another way you can create database objects is through the Control Center. The Control Center can be used instead of the SQL statements, command line processor commands, or APIs.

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 *Quick Beginnings* manual and operating system documentation for precise differences.

As an example, Windows supports an application type known as a “service”. DB2 for Windows will have each 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 Windows 32-bit application that uses the service functions included in the Microsoft Windows 32-bit application programming interface (API). Services can execute even when no user is logged on to the system.

Unless specifically mentioned, references to Windows 9x will refer to Windows 98, and Windows ME. References to Windows NT will refer to Windows NT, Windows

2000, Windows XP, and Windows Server 2003. References to Windows will mean all supported Windows operating systems.

Working with instances

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

- "Starting DB2 UDB on UNIX"
- "Starting DB2 UDB on Windows" on page 5
- "Multiple instances of the database manager" on page 5
- "Grouping objects by schema" on page 7
- "Parallelism" on page 8
- "Enabling data partitioning in a database" on page 11
- "Stopping an instance on UNIX" on page 12

Starting DB2 UDB on UNIX

You may need to start or stop DB2 Universal Database™ (DB2 UDB) 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.

Prerequisites:

To start a DB2 UDB 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. 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.

Procedure:

Use one of these two methods to start the instance:

1. 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.

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

```
db2start
```

Related tasks:

- "Stopping an instance on UNIX" on page 12
- "Setting the current instance" on page 24
- "Starting DB2 UDB on Windows" on page 5

Starting DB2 UDB on Windows

You may need to start or stop DB2 Universal Database™ (DB2 UDB) 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.

Prerequisites:

In order to successfully launch DB2 UDB as a service from **db2start**, the user account must have the correct privilege as defined by the Windows NT operating system to start a Windows service. The user account can be a member of the Administrators, Server Operators, or Power Users group.

Procedure:

Use one of these two methods to start the instance:

1. 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.

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

   ```
   db2start
   ```

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

When running in a partitioned database environment, each database partition server is started as a Windows service. You can not use the ″/D″ switch to start DB2 as a process in a partitioned database environment.

Related tasks:

- “Starting DB2 UDB on UNIX” on page 4
- “Stopping an instance on UNIX” on page 12
- “Stopping an instance on Windows” on page 13

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.

Note: (On UNIX® operating systems only:) To prevent environmental conflicts between two or more instances, you should ensure that each instance has its own home filesystem. Errors will be returned when the home file system is shared.

DB2® Universal Database (DB2 UDB) 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.

Related concepts:

- "Multiple instances on a UNIX operating system" on page 18
- "Multiple instances on a Windows operating system" on page 19

Related tasks:

- "Creating additional instances" on page 20

Related reference:

- "ATTACH Command" in the *Command Reference*

Attaching to another instance of the database manager

To attach to another instance, which may be remote, use the **ATTACH** command.

Prerequisites:

More than one instance must already exist.

Procedure:

To attach to another instance of the database manager using 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 to an 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 running the following command:

```
db2 detach
```

Related reference:

- "ATTACH Command" in the *Command Reference*
- "DETACH Command" in the *Command Reference*

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.

Some objects are created within certain schemas and stored in the system catalog tables 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.

Related concepts:

- "Definition of system catalog tables" on page 65

Related tasks:

- "Creating a schema" on page 81

Related reference:

- "SET SCHEMA statement" in the *SQL Reference, Volume 2*
- "CURRENT SCHEMA special register" in the *SQL Reference, Volume 1*

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 inter-partition query parallelism

Procedure:

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

Related concepts:

- "Partition and processor environments" in the *Administration Guide: Planning*
- "Data partitioning" in the *Administration Guide: Planning*
- "Database partition group design" in the *Administration Guide: Planning*
- "Partitions in a partitioned database" in the *Administration Guide: Performance*

Related tasks:

- "Enabling intra-partition parallelism for queries" on page 8
- "Enabling data partitioning in a database" on page 11
- "Redistributing data across partitions" in the *Administration Guide: Performance*

Enabling intra-partition parallelism for queries

Procedure:

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.

In order for intra-partition query parallelism to occur, you must modify one or more database configuration parameters, database manager configuration parameters, precompile or bind options, or a special register.

intra_parallel
: Database manager configuration parameter that specifies whether the database manager can use intra-partition parallelism. The default is not to use intra-partition parallelism.

max_querydegree
Database manager configuration parameter that specifies the maximum degree of intra-partition parallelism that is used for any SQL statement running on this instance. An SQL statement will not use more than the number given by this parameter when running parallel operations within a partition. The *intra_parallel* configuration parameter must also be set to "YES" for the value in *max_querydegree* is used. The default value for this configuration parameter is -1. This value means that the system uses the degree of parallelism determined by the optimizer; otherwise, the user-specified value is used.

dft_degree
Database configuration parameter. Provides the default for the DEGREE bind option and the CURRENT DEGREE special register. The default value is 1. A value of ANY means the system uses the degree of parallelism determined by the optimizer.

DEGREE
Precompile or bind option for static SQL.

CURRENT DEGREE
Special register for dynamic SQL.

Related concepts:

- "Parallel processing for applications" in the *Administration Guide: Performance*
- "Parallel processing information" in the *Administration Guide: Performance*

Related tasks:

- "Configuring DB2 with configuration parameters" in the *Administration Guide: Performance*

Related reference:

- "max_querydegree - Maximum query degree of parallelism configuration parameter" in the *Administration Guide: Performance*
- "intra_parallel - Enable intra-partition parallelism configuration parameter" in the *Administration Guide: Performance*
- "dft_degree - Default degree configuration parameter" in the *Administration Guide: Performance*
- "BIND Command" in the *Command Reference*
- "PRECOMPILE Command" in the *Command Reference*
- "CURRENT DEGREE special register" in the *SQL Reference, Volume 1*

Enabling intra-partition parallelism for utilities

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

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

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

Enabling parallelism for loading data: The load utility automatically makes use of parallelism, or you can use the following parameters on the **LOAD** command:

- CPU_PARALLELISM
- DISK_PARALLELISM

In a partitioned database environment, inter-partition parallelism for data loading occurs automatically when the target table is defined on multiple partitions. Inter-partition parallelism for data loading can be overridden by specifying OUTPUT_DBPARTNUMBS. The load utility also intelligently enables data partitioning parallelism depending on the size of the target partitions. MAX_NUM_PART_AGENTS can be used to control the maximum degree of parallelism selected by the Load utility. Data partitioning parallelism can be overridden by specifying PARTITIONING_DBPARTNUMS when ANYORDER is also specified.

Related concepts:

- "Load Overview" in the *Data Movement Utilities Guide and Reference*
- "Partitioned database load - overview" in the *Data Movement Utilities Guide and Reference*

Enabling parallelism when creating indexes: 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.

Related reference:

- "intra_parallel - Enable intra-partition parallelism configuration parameter" in the *Administration Guide: Performance*
- "CREATE INDEX statement" in the *SQL Reference, Volume 2*

Enabling I/O parallelism when backing up a database or 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 by defining multiple containers, or use a single container with multiple disks, and the appropriate use of the DB2_PARALLEL_IO registry variable. If you want to take advantage of parallel I/O, you must consider the implications of what must be done before you define any containers. This cannot be done whenever you see a need; it must be planned for before you reach the point where you need to backup your database or table space.
- 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.

Related reference:

- "BACKUP DATABASE Command" in the *Command Reference*

Enabling I/O parallelism when restoring a database or 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. You must make the decision to use this option before you define your containers. This cannot be done whenever you see a need; it must be planned for before you reach the point where you need to restore your database or table space.
- 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.

Related reference:

- "RESTORE DATABASE Command" in the *Command Reference*

Enabling data partitioning in a database

The decision to have your database working in a partitioned environment must be made before you create your database. As part of the database design decisions you make, you will have to determine if you should take advantage of the performance improvements through partitioning your database.

Some of the considerations surrounding your decision to create a partitioned database are made here.

Procedure:

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).

Before creating a partitioned database, you must select which database partition will be the catalog node for the database. You can then create the database directly from that partition, or from a remote client that is attached to that partition. 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. When working on UNIX, the system database directory is `sqldbdir` and is located in the `sqllib` directory under your home directory, or under the directory where DB2 Universal Database™ (DB2 UDB) was installed. When working on UNIX, 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. When working on Windows, the system database directory is located in the instance directory.

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*.

Related tasks:

- "Configuring DB2 with configuration parameters" in the *Administration Guide: Performance*

Related reference:

- "sqlecrea - Create Database" in the *Administrative API Reference*
- "CREATE DATABASE Command" in the *Command Reference*

Stopping an instance on UNIX

You may need to stop the current instance of the database manager.

Prerequisites:

To stop an 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.

Restrictions:

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 the instance 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.

Procedure:

Use one of these two methods to stop the instance:

1. To stop the instance using the Control Center:

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

2. To stop the instance using the command line, enter:

   ```
   db2stop
   ```

You can use the db2stop command to stop, or drop, individual partitions within a partitioned database environment. When working in a partitioned database and you are attempting to drop a logical partition using

```
db2stop drop nodenum <0>
```

you must ensure that no users are attempting to access the database. If they are, you will receive an error message SQL6030N.

Related reference:

- "db2stop - Stop DB2 Command" in the *Command Reference*
- "TERMINATE Command" in the *Command Reference*

Stopping an instance on Windows

You may need to stop the current instance of the database manager.

Prerequisites:

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

1. The user account stopping the DB2 Universal Database™ (DB2 UDB) service must have the correct privilege as defined by the Windows operating system. The user account can be a member of the Administrators, Server Operators, or Power Users group.
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.

Restrictions:

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 UDB 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.

Procedure:

To stop an instance on your system, use one of the following methods:

- **db2stop**
- Stop the service using the Control Center

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

- Stop using the "NET STOP" command.
- Stop the instance from within an application.

Recall that when you are using DB2 UDB in a partitioned database environment, each database partition server is started as a service. Each service must be stopped.

Related reference:

- "db2stop - Stop DB2 Command" in the *Command Reference*

Preparing to create a database

There are many concepts and tasks you should consider as part of the work to be done before you actually create a database. These concepts and tasks include designing your database and establishing the instance, the directories, and the other support files needed to work with a database. These topics are covered here:

- Designing logical and physical database characteristics
- Instance creation
- Environment variables and the profile registry
- DB2 Administration Server

- Creating a node configuration file
- Creating the database configuration file
- Fast communications manager (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*.

Instance creation

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 node configuration file (`db2nodes.cfg`)
- Any other files that contain debugging information, such as the exception or register dump or the call stack for the DB2® Universal Database (DB2 UDB) 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.

On Windows® operating systems, the instance directory is located in the `/sqllib` sub-directory, in the directory where DB2 UDB was installed.

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 UDB 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.

On Windows, the instance name is the same as the name of the service, so it should not conflict. You must have the correct authorization to create a service.

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 UDB, the database environment for each user must be updated so that it can access an instance and run the DB2 UDB 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.

Related concepts:

- “Multiple instances on a UNIX operating system” on page 18
- “Multiple instances on a Windows operating system” on page 19

Related tasks:

- “Add an instance” on page 23
- “UNIX details when creating instances” on page 21
- “Windows details when creating instances” on page 22
- “Setting the current instance” on page 24
- “Auto-starting instances” on page 24
- “Running multiple instances concurrently” on page 25
- “Listing instances” on page 23
- “Creating additional instances” on page 20

Setting the DB2 UDB environment automatically on UNIX

By default, the scripts that set up the database environment when you create an instance 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.

Procedure:

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.

Related tasks:

- "Setting the DB2 UDB environment manually on UNIX" on page 17

Setting the DB2 UDB environment manually on UNIX

Procedure:

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 CLP executable. Verify that it is located under the instance's `sqllib` directory.

Related tasks:

- "Setting the DB2 UDB environment automatically on UNIX" on page 17

Multiple instances on a UNIX operating system

It is possible to have more than one instance on a UNIX® operating system. However, you may only work within one instance of DB2® Universal Database (DB2 UDB) at a time.

Note: To prevent environmental conflicts between two or more instances, you should ensure that each instance has its own home filesystem. Errors will be returned when the home filesystem is shared.

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 UDB.

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.

Related tasks:

- "UNIX details when creating instances" on page 21

Multiple instances on a Windows operating system

It is possible to run multiple instances of DB2® Universal Database (DB2 UDB) on the same machine. Each instance of DB2 UDB maintains its own databases and has its own database manager configuration parameters.

An instance of DB2 UDB consists of the following:

- A Windows® service that represents the instance. The name of the service is same as the instance name. The display name of the service (from the Services panel) is the instance name, prefixed with the "DB2 - " string. For example, for an instance named DB2, there exists a Windows service called "DB2" with a display name of "DB2 - DB2".

  Note: A Windows service is not created for Windows 98, for Windows ME, or for client instances.
- An instance directory. This directory contains the database manager configuration files, the system database directory, the node directory, the DCS database directory, all the diagnostic log and dump files that are associated with the instance. The instance directory is by default a sub-directory inside the SQLLIB directory and has the same name as the instance name. For example, the instance directory for instance "DB2" is `C:\SQLLIB\DB2`, where `C:\SQLLIB` is where DB2 UDB is installed. You can use the registry variable DB2INSTPROF to change the default location of the instance directory. If the DB2INSTPROF registry variable is set to another location, then the instance directory is created under the directory pointed to by DB2INSTPROF. For example, if `DB2INSTPROF=D:\DB2PROFS`, then the instance directory will be `D:\DB2PROFS\DB2`.
- A registry key under HKEY_LOCAL_MACHINE\SOFTWARE\IBM®\DB2\PROFILES\<instance_name>. All the instance level registry variables are created here.

You can run multiple DB2 UDB instances concurrently. To work with an instance, you need to set the DB2INSTANCE environment variable to the name of the instance before issuing commands against that instance.

To prevent one instance from accessing the database of another instance, the database files for an instance are created under a directory that has the same name as the instance name. For example, when creating a database on drive `C:` for instance DB2, the database files are created inside a directory called `C:\DB2`. Similarly, when creating a database on drive `C:` for instance TEST, the database files are created inside a directory called `C:\TEST`.

Related concepts:

- "High availability" in the *Data Recovery and High Availability Guide and Reference*

Related tasks:

- "Windows details when creating instances" on page 22

Creating additional instances

Although an instance is created as part of the installation of DB2 Universal Database™ (DB2 UDB), your business needs may require you to create additional instances.

Prerequisites:

If you belong to the Administrative group on Windows, or you have root authority on UNIX platforms, you can add additional DB2 UDB instances. The machine where you add the instance becomes the instance-owning machine (node zero). Ensure that you add instances on a machine where a DB2 administration server resides.

Procedure:

To add an instance using the command line, enter:

```
db2icrt <instance_name>
```

When using the **db2icrt** command to add another instance of DB2 UDB, 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.

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.

For DB2 Universal Database Enterprise – Server Edition (ESE) , you also need to declare that you are adding a new instance that is a partitioned database system. In addition, when working with a ESE instance having more than one partition, and working with Fast Communication Manager (FCM), you can have multiple connections between partitions by defining more TCP/IP ports when creating the instance. For example, for Windows operating systems, use the **db2icrt** command with the **-r <port range>** parameter. The port range is shown as follows:

```
-r:<base_port,end_port>
```

where the base_port is the first port that can be used by FCM, and the end_port is the last port in a range of port numbers that can be used by FCM.

Related concepts:

- "Authentication methods for your server" on page 207
- "Authentication considerations for remote clients" on page 212

Related reference:

- "db2icrt - Create Instance Command" in the *Command Reference*

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, SERVER_ENCRYPT, or CLIENT. If not specified, the default is SERVER, if a DB2 Universal Database™ (DB2 UDB) 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 UDB client or the DB2 UDB Application Development Client. For other DB2 UDB 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 created. Valid instance types are: `ese`, `wse`, `client`, and `standalone`.

Examples:

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

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

DB2 UDB 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.

Related reference:

- “db2icrt - Create Instance Command” in the *Command Reference*

Windows details when creating instances

When working with the Windows operating systems, the **db2icrt** command has the following optional parameters:

- –s InstType

  Allows different types of instances to be created. Valid instance types are: `ese`, `wse`, `client`, and `standalone`.

- –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 domain/user account name and password of the DB2 Universal Database 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.

The following example could be used, on DB2 Universal Database (DB2 UDB) Enterprise Server Edition for Windows:

```
db2icrt inst1 –s ese
   –p:\\machineA\db2mpp
   –u:<user account name>,<password> –r:9010,9015
```

Note: If you change the service account; that is, if you no longer use the default service created when the first instance was created during product installation, then you must grant the domain/user account name used to create the instance the following advanced rights:

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

The instance requires these user rights to access the shared drive, authenticate the user account, and run DB2 UDB as a Windows service. The "Lock page in memory" right is needed for Address Windowing Extensions (AWE) support.

Related reference:

- "db2icrt - Create Instance Command" in the *Command Reference*

Add an instance

Procedure:

Once you have created an additional instance, you will need to add a record of that instance within the Control Center to be able to work with that instance from the Control Center.

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:

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

Related concepts:

- "Instance creation" on page 15

Related tasks:

- "Listing instances" on page 23

Listing instances

Procedure:

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 **Instances** folder.
2. Right-click the Instances folder, and select **Add** from the pop-up menu.
3. On the Add Instance window, click **Refresh**.
4. Click the drop-down arrow to see a list of database instances.
5. Click **Cancel** to exit the window.

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 supported Windows platforms) use:

```
set db2instance
```

Related reference:

- "db2ilist - List Instances Command" in the *Command Reference*

Setting the current instance

Procedure:

When you run commands to start or stop an instance's database manager, DB2 Universal Database™ (DB2 UDB) applies the command to the current instance. DB2 UDB 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 UDB 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 9x, they are set in the `autoexec.bat` file.
- If the DB2INSTANCE environment variable is not set at all, DB2 UDB 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
  ```

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

```
db2 get instance
```

Related tasks:

- "Declaring registry and environment variables" on page 28

Auto-starting instances

Procedure:

On Windows operating systems, the DB2 Universal Database™ (DB2 UDB) instance that is created during install is set as auto-started by default. An instance created using **db2icrt** is set as a manual start. To change the start type, you need to go to the Services panel and change the property of the DB2 UDB service there.

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

```
db2iauto -on <instance name>
```

where `<instance name>` 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 <instance name>
```

where `<instance name>` is the login name of the instance.

Related concepts:

- "Instance creation" on page 15

Related reference:

- "db2iauto - Auto-start Instance Command" in the *Command Reference*

Running multiple instances concurrently

Procedure:

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.

(On Windows only:) 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.

Related concepts:

- "Multiple instances of the database manager" on page 5

Related tasks:

- "UNIX details when creating instances" on page 21
- "Windows details when creating instances" on page 22
- "Creating additional instances" on page 20

License management

The management of licenses for your DB2® Universal Database (DB2 UDB) 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.

When the Control Center cannot be used, the **db2licm** Licensed Management Tool command performs basic license functions. With this command, you are able to add, remove, list, and modify licenses and policies installed on your local system.

Related reference:

- "db2licm - License Management Tool Command" in the *Command Reference*

Environment variables and the profile registry

Environment and registry variables control your database environment.

You can use the Configuration Assistant (**db2ca**) to configure configuration parameters and registry variables.

Prior to the introduction of the DB2® Universal Database (DB2 UDB) profile registry, changing your environment variables on Windows® 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 UDB profile registries. Users on UNIX® operating systems with system

administration (SYSADM) authority for a given instance can update registry values for that instance. Windows users do not need SYSADM authority to update registry variables. Use the **db2set** command to update registry variables without rebooting; this information is stored immediately in the profile registries. The DB2 UDB registry applies the updated information to DB2 UDB server instances and DB2 UDB applications started after the changes are made.

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

Note: There are DB2 UDB environment variables `DB2INSTANCE`, and `DB2NODE` which may not be stored in the DB2 UDB profile registries. On some operating systems the **set** command must be used in order to update these environment variables. 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. Different levels of support are now provided through the different profiles. Remote administration of the environment variables is also available when using the DB2 Administration Server.

There are four profile registries:

- The DB2 UDB Instance Level Profile Registry. The majority of the DB2 UDB 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 UDB 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 ESE, one global-level profile exists at each machine.
- The DB2 UDB Instance Node Level Profile Registry. This registry level contains variable settings that are specific to a partition (node) in a multi-partition environment. Values defined in this level override their settings at the instance and global levels.
- The DB2 UDB Instance Profile Registry. This registry contains a list of all instance names recognized by this system. You can see the complete list of all the instances available on the system by running `db2ilist`.

DB2 UDB 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 <instance name> <nodenum>** command).
3. Registry values set with the instance level profile (using the **db2set -i** command).
4. Registry values set with the global level profile (using the **db2set -g** command).

Instance Level Profile Registry

There are a couple of UNIX and Windows differences when working with a partitioned database environment. These differences are shown in the following example.

Assume that there is a partitioned database environment with three physical nodes that we will identify as “red”, “white”, and “blue”. On UNIX platforms, if the instance owner runs the following from any of the nodes:

```
db2set -i FOO=BAR
```

or

```
db2set FOO=BAR        ('-i' is implied)
```

the value of FOO will be visible to all nodes of the current instance (that is, “red”, “white”, and “blue”).

On UNIX platforms, the instance level profile registry is stored in a text file inside the `sqllib` directory. In partitioned database environments, the `sqllib` directory is located on the filesystem shared by all physical nodes.

On Windows platforms, if the user performs the same command from “red”, the value of FOO will only be visible on “red” of the current instance. DB2 UDB stores the instance level profile registry inside the Windows registry. There is no sharing across physical nodes. To set the registry variables on all the physical machines, use the “rah” command as follows:

```
rah db2set -i FOO=BAR
```

rah will remotely run the db2set command on “red”, “white”, and “blue”.

It is possible to use DB2REMOTEPREG so that the registry variables on non-instance-owning machines are configured to refer to those on the instance owning machine. This effectively creates an environment where the registry variables on the instance-owning machine are shared amongst all machines in the instance.

Using the example shown above, and assuming that “red” is the owning machine, then one would set DB2REMOTEPREG on “white” and “blue” machines to share the registry variables on “red” by doing the following:

```
(on red) do nothing
(on white and blue) db2set DB2REMOTEPREG=\\red
```

The setting for DB2REMOTEPREG must not be changed after it is set.

Here is how REMOTEPREG works:

When DB2 UDB reads the registry variables on Windows, it first reads the DB2REMOTEPREG value. If DB2REMOTEPREG is set, it then opens the registry on the remote machine whose machine name is specified in the DB2REMOTEPREG variable. Subsequent reading and updating of the registry variables will be redirected to the specified remote machine.

Accessing the remote registry requires that the Remote Registry Service is running on the target machine. Also, the user logon account and all DB2 UDB service logon accounts have sufficient access to the remote registry. Therefore, to use DB2REMOTEPREG, you should operate in a Windows domain environment so that the required registry access can be granted to the domain account.

There are Microsoft® Cluster Server (MSCS) considerations. You should not use DB2REMOTEPREG in an MSCS environment. When running in an MSCS configuration where all machines belong to the same MSCS cluster, the registry

variables are maintained in the cluster registry. Therefore, they are already shared between all machines in the same MSCS cluster and there is no need to use DB2REMOTEPREG in this case.

When running in a multi-partitioned failover environment where partitions span across multiple MSCS clusters, you cannot use DB2REMOTEPREG to point to the instance-owning machine because the registry variables of the instance-owning machine reside in the cluster registry.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related tasks:

- "Declaring registry and environment variables" on page 28

Declaring registry and environment variables

Procedure:

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 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 a particular partition in an instance, use:

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

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

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

If you user the Lightweight Directory Access Protocol (LDAP), you can set registry variables in LDAP using:

- To set registry variables at the user level within LDAP, use:

  ```
  db2set -ul
  ```

- To set registry variables at the global level within LDAP, use:

  ```
  db2set -gl user_name
  ```

When running in an LDAP environment, it is possible to set a DB2 Universal Database™ (DB2 UDB) 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, there are only two DB2 UDB registry variables that can be set at the LDAP global level: DB2LDAP_KEEP_CONNECTION and DB2LDAP_SEARCH_SCOPE.

For example, 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”.

Notes:

1. When the DB2 UDB profile.env file is updated simultaneously (that is, at the same time or very close to the same time) by the db2set command, the size of the profile.env file is reduced to zero. Also, the output from `db2set -all` displays inconsistent values.
2. There is a difference between the `-g` option which is used to set DB2 UDB registry variables at the machine global level and the `-gl` which is specific to the LDAP global level.
3. The user level registry variable is only supported on Windows when running in an LDAP environment.
4. Variable settings at the user level contains user specific variable settings. Any changes to the user level are written to the LDAP directory.
5. The parameters “-i”, “-g”, “-gl”, and “-ul” cannot be used at the same time in the same command.
6. Some variables will always default to the global level profile. They cannot be set at the instance or node level profiles; for example, DB2SYSTEM and DB2INSTDEF.
7. 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.

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
```

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 UDB global-level profile). However, the variable you specify could still be set by a lower level profile (in this case, the DB2 UDB node-level profile).

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related tasks:

- "Setting environment variables on Windows" on page 30
- "Setting environment variables on UNIX systems" on page 32
- "Searching the LDAP directory partitions or domains" on page 317
- "Setting DB2 registry variables at the user level in the LDAP environment" on page 319

Setting environment variables on Windows

Procedure:

It is strongly recommended that all specific registry variables be defined in the DB2 Universal Database™ (DB2 UDB) profile registry. If DB2 UDB 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 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 UDB 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 UDB Enterprise Server Edition servers on Windows 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 UDB 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 UDB profile registry. If this variable is not set, the target logical node defaults to the logical node which is defined as 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 9x: Edit the `autoexec.bat` file, and reboot the system to have the change take effect.

On Windows: You can set the DB2 UDB environment variables DB2INSTANCE and DB2NODE as follows (using DB2INSTANCE in this description):

- (On Windows NT and Windows 2000) Select **Start**, **Settings**, **Control Panel**. (On Windows XP and Windows Server 2003) Select **Start** —> **Control Panel**.
- (On Windows NT and Windows 2000) Double-click on the **System** icon. (On Windows XP and Windows Server 2003) Depending on the Windows theme and the currently selected view type, you may have to select **Performance** and **Maintenance** before you can select the **System** icon.
- (On Windows NT)In the System Control Panel, in the System Environment Variables section, do the following: (On Windows 2000, Windows XP, and Windows Server 2003) From the System Properties window you have to select the **Advanced** tab and then click on the Environment Variables button and do the following:
  1. If the DB2INSTANCE variable does not exist:
     a. (On Windows NT) Select any system environment variable. (On Windows 2000, Windows XP, and Windows Server 2003) Click on the **New** button.
     b. (On Windows NT) Change the name in the *Variable* field to DB2INSTANCE. (On Windows 2000, Windows XP, and Windows Server 2003) Fill in the *Variable Name* field with DB2INSTANCE.
     c. (On Windows NT) Change the *Value* field to the instance name, for example db2inst. (On Windows 2000, Windows XP, and Windows Server 2003) Fill in the *Variable Value* field with 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. (On Windows NT) Select Set. (On Windows 2000, Windows XP, and Windows Server 2003) Select OK.
  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 UDB instance called TEST, issue the following commands in a command window:

```
set DB2INSTANCE=TEST
db2start
```

When working in C Shell, issue the following commands in a command window:

```
setenv DB2INSTANCE TEST
```

The profile registries are located as follows:

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

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

  Note: The *instance_name* is the name of the DB2 UDB instance.
- The DB2 UDB Global Level Profile Registry in the Windows registry, with the path:

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

- The DB2 UDB Instance Node Level Profile Registry in the Windows 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.
- There is no DB2 UDB Instance Profile Registry required. For each of the DB2 UDB instances in the system, a key is created in the path:

  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES*instance_name*

  The list of instances can be obtained by counting the keys under the PROFILES key.

Related concepts:

- "DB2 Administration Server" on page 39

Related tasks:

- "Setting environment variables on UNIX systems" on page 32

Setting environment variables on UNIX systems

Procedure:

It is strongly recommended that all specific registry variables be defined in the DB2 UDB profile registry. If DB2 UDB 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 supported variables must be set in the DB2 UDB profile registry. To set variables that are not supported by DB2 UDB, define them in your script files, `userprofile` and `usercshrc`.

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:

```
export DB2INSTANCE=<inst1>
```

- For C shell:

```
setenv DB2INSTANCE <inst1>
```

In order for the DB2 UDB profile registry to be administered properly, the following file ownership rules must be followed on UNIX operating systems.

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

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

  The access permissions and ownership of this file should be:

  ```
  -rw-rw-r-- <db2inst1> <db2iadm1> profile.env
  ```

  where <db2inst1> is the instance owner, and <db2iadm1> is the instance owner's group.

  The INSTHOME is the home path of the instance owner.

- The DB2 UDB Global Level Profile Registry is located under:
  - /var/db2/<version_id>/default.env for AIX, Solaris Operating Environment, and Linux 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-rw-r-- <Instance_Owner> <Instance_Owner_Group> default.env
  ```

  In order to modify a global registry variables, a user must be logged on as: root.

- The DB2 UDB 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:

  ```
  drwxrwsr-w <Instance_Owner> <Instance_Owner_Group> nodes

  -rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> <node_number>.env
  ```

  The INSTHOME is the home path of the instance owner.

- The DB2 UDB Instance Profile Registry is located under:
  - /var/db2/<version_id>/profiles.reg for AIX, Solaris, and Linux 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
  ```

Related concepts:

- "DB2 Administration Server" on page 39

Related tasks:

- "Setting environment variables on Windows" on page 30

Creating a node configuration file

Procedure:

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 Considerations

If you are using DB2 Universal Database™ (DB2 UDB) Enterprise - Server Edition on Windows, 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. You can use the **db2ncrt** command to add a database partition server to an instance. You can use the **db2ndrop** command to drop a database partition server to an instance. You can use the **db2nchg** command to modify a database partition server configuration including moving the database partition server from one machine to another; changing the TCP/IP host name; or, selecting a different logical port or network name.

Note: You should not create files or directories under the `sqllib` subdirectory other than those created by DB2 UDB 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. The other exception is when user-defined 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:

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

Tokens are delimited by blanks. The variables are:

dbpartitionnum

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

Once a database partition 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 database partition number can be used again for any new node that you add.

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

```
NODEnnnn
```

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

hostname

The hostname of the IP address for inter-partition communications. Use the fully-qualified name for the hostname. The `/etc/hosts` file also should use the fully-qualified name. If the fully-qualified name is not used in the db2nodes.cfg file and in the `/etc/hosts` file, you may receive error message SQL30082N RC=3.

(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 dbpartitionnum for a host), assign the logical-port numbers to the logical nodes from 0, with no gaps. Order is not important.

For example, the following setup is valid:

```
0   cpaiss43.mach1.xxx.com   1
1   cpaiss43.mach1.xxx.com   0
2   cpaiss43.mach1.xxx.com   2
3   cpaiss44.mach1.xxx.com   0
```

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 partitions, and uses SP2SW1 as the DB2 UDB interface. It also shows the partition numbers starting at 1 (rather than at 0), and a gap in the dbpartitionnum sequence:

Table 1. Database partition number example table.

dbpartitionnum	hostname	logical-port	netname
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

You can update the db2nodes.cfg file using an editor of your choice. (The exception is: an editor should not be used on Windows.) You must be careful, however, to protect the integrity of the information in the file, as data partitioning requires that the database partition 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.

Related concepts:

- "Guidelines for stored procedures" in the *Administration Guide: Performance*

Related reference:

- "db2start - Start DB2 Command" in the *Command Reference*
- "db2stop - Stop DB2 Command" in the *Command Reference*
- "CREATE DATABASE Command" in the *Command Reference*
- "DROP DATABASE Command" in the *Command Reference*
- "db2nchg - Change Database Partition Server Configuration Command" in the *Command Reference*
- "db2ncrt - Add Database Partition Server to an Instance Command" in the *Command Reference*
- "db2ndrop - Drop Database Partition Server from an Instance Command" in the *Command Reference*

Creating the database configuration file

Procedure:

A *database configuration file* is 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 or used when creating the database (for example, database code page, collating sequence, DB2 UDB 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).

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.

Related concepts:

- "Issuing commands in a partitioned database environment" on page 343

Related tasks:

- "Configuring DB2 with configuration parameters" in the *Administration Guide: Performance*

Fast communications manager (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® Considerations

If you are using DB2® Universal Database (DB2 UDB) Enterprise - Server Edition in the Windows 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

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 instancev 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 `services` file of the `etc` directory 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 `services` file of the `etc` directory is not shared, the same considerations apply, with one additional consideration: you must ensure that the entries defined for the DB2 UDB instance are the same in all `services` files of the `etc` directory (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 `services` file of the `etc` directory 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. For example,

```
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.

Related concepts:

- "Partition and processor environments" in the *Administration Guide: Planning*

Chapter 2. Creating and using the DB2 Administration Server (DAS)

The DB2 administration server (DAS) is used to assist with DB2 server tasks.

DB2 Administration Server

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

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

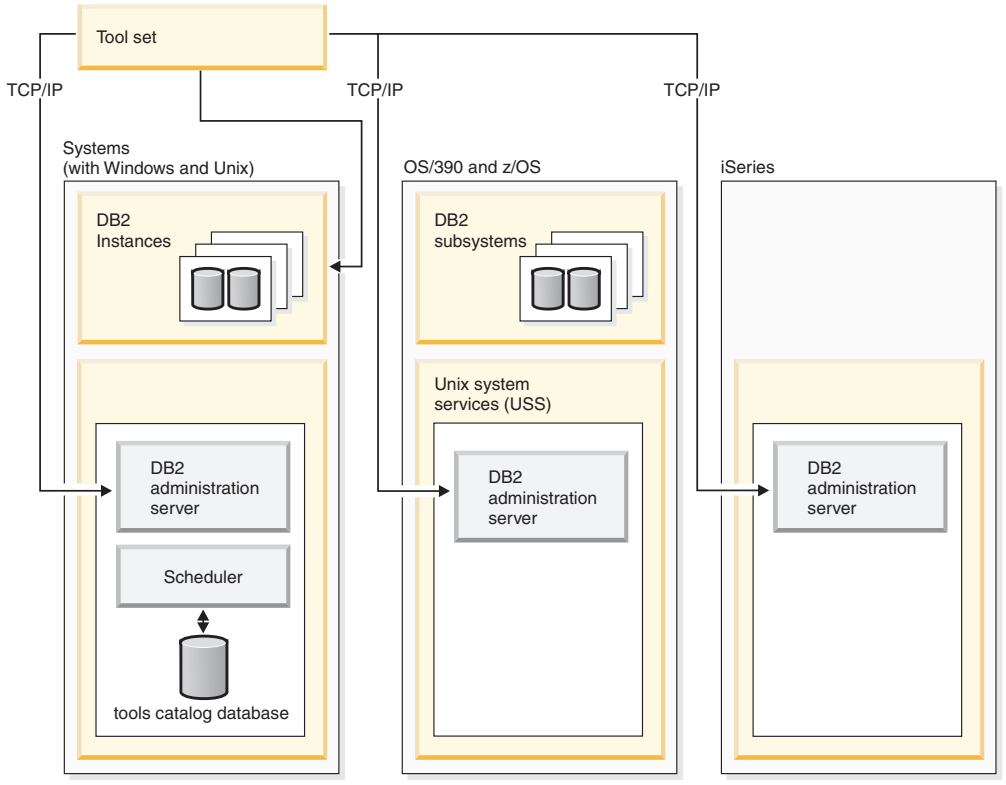


Figure 1. Where DAS is used

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

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

The DAS is available on all supported Windows® and UNIX® platforms as well as the zSeries® (OS/390 and z/OS™ only) platforms. The DAS on zSeries is used to support the Control Center, Development Center, and Replication Center in administrative tasks.

The DB2 administration server on zSeries (OS/390 and z/OS only), will be packaged and delivered as part of the DB2 Management clients feature of DB2 UDB. Products that need DAS, like the Control Center, Replication Center, and Development Center, require the installation of the DAS function. For information on the availablility of DAS on your operating system, contact your IBM® representative.

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

Although a scheduler is not provided on zSeries (OS/390 and z/OS only), you can use the Build JCL and Create JCL functions provided in the Control Center to generate JCL that is saved in partitioned datasets to be run using your system scheduler.

Related concepts:

- "Security considerations for the DAS on Windows" on page 50
- "DAS configuration on Enterprise Server Edition (ESE) systems" on page 54
- "Discovery of administration servers, instances, and databases" on page 55
- "DB2 administration server first failure data capture" on page 59

Related tasks:

- "Create a DB2 Administration Server" on page 41
- "Starting and stopping the DAS" on page 42
- "Listing the DAS" on page 43
- "Configuring the DAS" on page 43
- "Updating the DAS on UNIX" on page 51
- "Removing the DAS" on page 51
- "Setting up DAS with Enterprise Server Edition (ESE) systems" on page 52
- "Hiding server instances and databases from discovery" on page 57
- "Setting discovery parameters" on page 57
- "Setting up the DAS to use the Configuration Assistant and the Control Center" on page 58
- "Update the DAS configuration for discovery" on page 59
- "Tools catalog database and DAS scheduler setup and configuration" on page 44
- "Notification and contact list setup and configuration" on page 49
- "DAS Java virtual machine setup" on page 49

Create a DB2 Administration Server

The DB2 Administration Server provides support services for DB2 Universal Database™ (DB2 UDB) tools such as the Control Center and the Configuration Assistant.

Prerequisites:

To create a DAS, you must have root authority on UNIX platforms or using an account that has the correct authorization to create a service.

On Windows, 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**.

Procedure:

Typically, the setup program creates a DAS on the instance-owning machine during DB2 UDB 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 Windows platforms:

  Log on to the machine you want to create the DAS on using an account that has the correct authorization to create a service.

  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.
- 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 DB2 UDB install path:

     ```
     dascrt -u <DASUser>
     ```

     `<DASUser>` is the user name of the DAS user you created when you were creating users and groups for DB2 UDB.

     – On AIX:

     ```
     /usr/opt/db2_08_01/instance/
     dascrt -u <DASUser>
     ```

     – On HP-UX, Solaris Operating Environment, or Linux:

     ```
     /opt/IBM/db2/V8.1/instance/
     dascrt -u <DASUser>
     ```

Related reference:

- "db2admin - DB2 Administration Server Command" in the *Command Reference*

Starting and stopping the DAS

Procedure:

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

To start or stop the DAS on Windows use the **db2admin start** or **db2admin stop** commands.

When working on DB2 Universal Database™ (DB2 UDB) 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:

     ```
     . DASHOME/das/dasprofile    (for Bourne or Korn shell)
     source DASHOME/das/dascshrc (for C shell)
     ```

     where `DASHOME` is the home directory of the DB2 administration server.
  3. To start the DAS use the **db2admin** command:

```
db2admin start
```

Note: The DAS is automatically started after each system reboot. The default startup behavior of the DAS can be altered using the **dasauto** command.

- To stop the DAS:
  1. Log in as an account or user ID that is part of the *dasadm_group*.
  2. 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. The user needs to belong to the *dasadm_group* to issue a **db2admin start** or **db2admin stop** command.

Related reference:

- "db2admin - DB2 Administration Server Command" in the *Command Reference*
- "dasadm_group - DAS administration authority group name configuration parameter" in the *Administration Guide: Performance*

Listing the DAS

Procedure:

To obtain the name of the DAS 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, and establish or modify the user account associated with the DAS.

Related reference:

- "db2admin - DB2 Administration Server Command" in the *Command Reference*

Configuring the DAS

Procedure:

To see the current values for the DB2 administration server 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.

In order to update the DAS configuration file using the Command Line Processor (CLP) and the UPDATE ADMIN CONFIG command, you must use the CLP from an instance that is at the same installed level as the DAS. To update individual entries in the DAS configuration file, enter:

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

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

```
db2 reset admin cfg
```

In some cases, changes to the DAS 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 platform, stopping and starting the service). In other cases, the configuration parameters are configurable online (that is, you do not have to restart the DAS for these to take affect).

Related tasks:

- "Configuring DB2 with configuration parameters" in the *Administration Guide: Performance*

Related reference:

- "UPDATE ADMIN CONFIGURATION Command" in the *Command Reference*

Tools catalog database and DAS scheduler setup and configuration

The tools catalog database contains task information created by the Task Center and Control Center. These tasks are run by the DB2 administration server's scheduler. The scheduler and the tools catalog database always work together; neither can function without the other. The scheduler is a specific piece of the DB2 administration server that acts as an agent to read the tools catalog database and runs the tasks at their respective times.

Prerequisites:

The DB2 administration server must be installed.

Procedure:

The goal is to set up and configure the tools catalog database and the DAS scheduler.

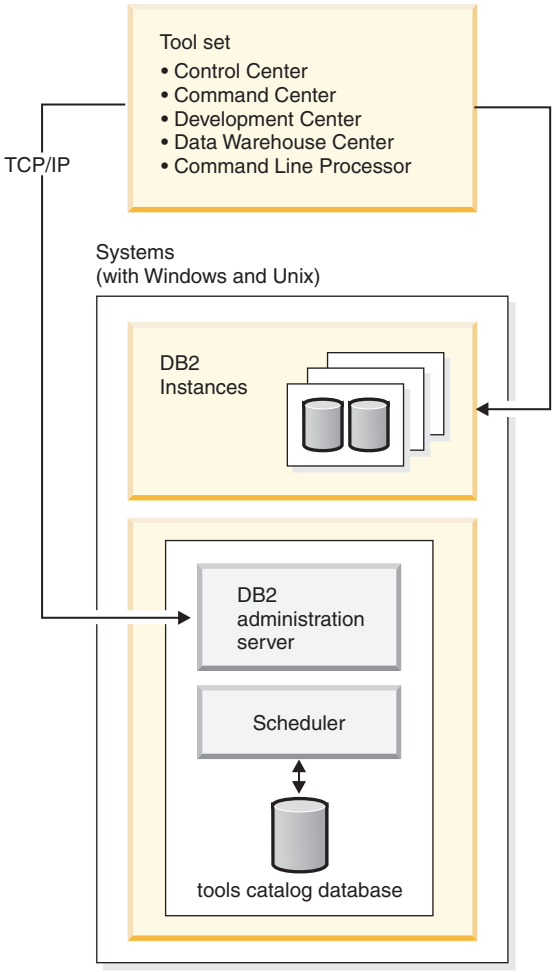


Figure 2. How DAS relates to other parts of DB2 UDB

The DB2 administration server Configuration process tells the Scheduler the location of the tools catalog database, and whether or not the Scheduler should be enabled. By default, when a tools catalog database is created, its corresponding DAS configuration is updated. That is, the Scheduler is configured and ready to use the new tools catalog; there is no need to restart the DAS.

The tools catalog database can be created on a server that is local or remote from the Scheduler system. If the tools catalog is created on a remote server, it must be cataloged at the scheduler tools catalog database instance (TOOLSCAT_INST). In addition, the scheduler user ID must be set by using the command **db2admin setschedid**, so that the scheduler can connect and authenticate with the remote catalog. The full syntax for the **db2admin** command is found in the *Command Reference*.

The DAS scheduler requires a Java virtual machine (JVM) to access the tools catalog information. The JVM information is specified using the `jdk_path` DB2 administration server configuration parameter of the DAS.

The `jdk_64_path` configuration parameter is required if you are creating a tools catalog against a 64-bit instance on one of the platforms that supports both 32- and 64-bit instances (AIX, Sun, and HP-UX).

The Control Center and Task Center access the tools catalog database directly from the client. The tools catalog database therefore needs to be cataloged at the client before the Control Center can make use of it. The Control Center provides the means to automatically retrieve information about the tools catalog database and create the necessary directory entries in the local node directory and database directory. The only communication protocol supported for this automatic cataloging is TCP/IP.

One of the DAS configuration parameters is called *exec_exp_task*. This parameter specifies whether or not the scheduler executes the tasks that have been scheduled in the past, but have not yet been run. The scheduler only detects expired tasks when it starts up.

For example, if you have a job scheduled to run every Saturday, and the scheduler is turned off on Friday and then restarted on Monday, the job scheduled for Saturday is now a job that is scheduled in the past. If *exec_exp_task* is set to "Yes", your Saturday job runs when the scheduler is restarted.

The other DAS configuration parameters required by the scheduler consist of identifying the tools catalog database and the Simple Mail Transfer Protocol (SMTP) server to be used for notification.

The following examples explain how these parameters are used:

- An example Windows server setup.
  1. The tools catalog database can be any name you wish. In this example, the tools catalog database is called "CCMD" and is created under the DB2 Universal Database™ (DB2 UDB) instance on server machine Host1 (tcp/ip hostname Host1). A schema name is used to uniquely identify a tools catalog within a given database. For the purposes of this example, assume the schema name is "CCADMIN".
  2. The instance called "DB2" is setup for TCP/IP communications using port number 50000 by using:

     ```
     db2set -i DB2 DB2COMM=TCPIP
     db2 update dbm cfg using svcename db2cDB2
     db2stop
     db2start
     ```

  3. The `db2cDB2` service name is defined in `%SystemRoot%\system32\drivers\etc\services`. That is, in `services` you will find a line:

     ```
     db2cDB2     50000/tcp     #connection port for the DB2 instance DB2
     ```

  4. The tools catalog is created using the CREATE TOOLS CATALOG command. This will create the tools catalog tables and views with a schema name corresponding to the catalog name in the specified database. The DB2 administration server configuration parameters are automatically updated and the scheduler is enabled and started.
  5. Assume that the SMTP server used for e–mail notifications is on machine Host2 (tcp/ip hostname Host2). This information is then specified to the DB2 administration server using:

     ```
     db2 update admin cfg using smtp_server Host2
     ```

     This may be done during the installation process. If it is done later, it needs to be manually specified to the DAS via a DB2 UDB Version 8 CLP command as shown above.

6. The IBM Software Development Kit (SDK) for Java on Windows is installed under %DB2PATH%\java\jdk. This should be already specified to the DAS. It can be verified, and set if necessary, using:

   ```
   db2 update admin cfg using jdk_path c:\SQLLIB\java\jdk
   ```

   This assumes DB2 UDB is installed under C:\SQLLIB.

   Note: If the DAS is going to be created by db2admin create, make sure you use the /USER and /PASSWORD options. The USER account is used by the scheduler process. Without it, the scheduler will not be started properly. The USER account should have SYSADM authority on the tools catalog instance.

   If the DAS is going to be created by db2admin create, and no /USER and /PASSWORD options are to be specified at that time, then you can update the USER information at a later time. This update is done on DAS by running the following commands:

   ```
   db2admin stop
   db2admin setid <user account ID> <password>
   db2admin start
   ```

- An example Windows client setup.
  1. Assume that the Control Center is running on client machine C1 (tcp/ip hostname C1).
  2. The DAS is cataloged as an administration server node in the local node directory using either the Configuration Assistant or the Control Center, or by using the command:

     ```
     db2 catalog admin tcpip node Host1 remote Host1 system Host1 ostype NT
     ```

  3. If the Task Center is started and the system Host1 is selected, the Task Center attempts to find the tools catalog database in the local directory. (The Control Center could be used in place of the Task Center.) If not found, it attempts to catalog the node and database using:

     ```
     db2 catalog  tcpip node <unique-node name>
       remote Host1 server 50000
       remote_instance DB2 system Host1 ostype NT
     db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
     ```

     If the automatic cataloging is unsuccessful, the database can be cataloged using the Configuration Assistant or the Control Center. The database will then be recognized and used by the Task Center.
- An example AIX server setup.
  1. The tools catalog database can be any name you wish. In this example, the tools catalog database is called “CCMD” and is created under the db2inst1 instance on server machine Host1 (tcp/ip hostname Host1). A schema name is used to uniquely identify a tools catalog within a given database. For the purposes of this example, assume the schema name is “CCADMIN”.
  2. The instance db2inst1 is setup for TCP/IP communications using port number 50000 by using:

     ```
     db2set -i DB2 DB2COMM=TCPIP
     db2 update dbm cfg using svcename xdb2inst
     db2stop
     db2start
     ```

  3. The xdb2inst service name is defined in /etc/services. That is, in services you will find a line:

     ```
     xdb2inst1     50000/tcp     #connection port for the DB2 instance db2inst1
     ```

  4. The tools catalog is created using the CREATE TOOLS CATALOG command. This will create the tools catalog tables and views with a schema name

corresponding to the catalog name in the specified database. The DB2 administration server configuration parameters are automatically updated and the scheduler is enabled and started.

5. Assume that the SMTP server used for e–mail notifications is on machine Host2 (tcp/ip hostname Host2). This information is then specified to the DB2 administration server using:

   ```
   db2 update admin cfg using smtp_server Host2
   ```

   This may be done during the installation process. If it is done later, it needs to be manually specified to the DAS via a DB2 UDB Version 8 CLP command as shown above.

6. The IBM Software Developer's Kit for Java (SDK) Version 1.3.1 on AIX is installed under `/sqllib/java/jdk`. This should be already specified to the DAS. It can be verified, and set if necessary, using:

   ```
   db2 update admin cfg using jdk_path /sqllib/java/jdk
   ```

- An example AIX client setup.

  1. Assume that the Control Center is running on client machine C1 (tcp/ip hostname C1).
  2. The DAS is cataloged as an administration server node in the local node directory using either the Configuration Assistant or the Control Center by using:

     ```
     db2 catalog admin tcpip node Host1 remote Host1 system Host1
       ostype AIX
     ```

  3. If the Task Center is started and the system Host1 is selected, the Task Center attempts to find the tools catalog database in the local directory. (The Control Center could be used in place of the Task Center.) If not found, it attempts to catalog the node and database using:

     ```
     db2 catalog tcpip node <unique-node name>
       remote Host1 server 50000
       remote_instance DB2 system Host1 ostype AIX
     db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
     ```

     If the automatic cataloging is unsuccessful, the database can be cataloged using the Configuration Assistant or the Control Center. The database will then be recognized and used by the Task Center.

Related reference:

- "svcename - TCP/IP service name configuration parameter" in the *Administration Guide: Performance*
- "sched_enable - Scheduler mode configuration parameter" in the *Administration Guide: Performance*
- "toolscat_inst - Tools catalog database instance configuration parameter" in the *Administration Guide: Performance*
- "toolscat_db - Tools catalog database configuration parameter" in the *Administration Guide: Performance*
- "toolscat_schema - Tools catalog database schema configuration parameter" in the *Administration Guide: Performance*
- "smtp_server - SMTP server configuration parameter" in the *Administration Guide: Performance*
- "jdk_path - Software Developer's Kit for Java installation path DAS configuration parameter" in the *Administration Guide: Performance*
- "exec_exp_task - Execute expired tasks configuration parameter" in the *Administration Guide: Performance*

Notification and contact list setup and configuration

E–mail and pager notifications from the DB2 administration server (DAS) can be local or remote. A contact list is required to ensure that notifications are sent to the correct hostname.

Procedure:

There are two DAS configuration parameters used to enable notifications by the scheduler or the health monitor.

The DAS configuration parameter *smtp_server* is used to identify the Simple Mail Transfer Protocol (SMTP) server used by the scheduler to send e–mail and pager notifications as part of task execution completion actions as defined through the Task Center, or by the health monitor to send alert notifications via e–mail or pager.

The DAS configuration parameter *contact_host* specifies the location where the contact information used by the scheduler and health monitor for notification is stored. The location is defined to be a DB2 administration server's TCP/IP hostname. Allowing *contact_host* to be located on a remote DAS provides support for sharing a contact list across multiple DB2 administration servers. This should be set for partitioned database environments to ensure a common contact list is used for all partitions. The contact list is stored in a flat file under the DAS directory. If *contact_host* is not specified, the DAS assumes the contact information is local.

Related reference:

- "smtp_server - SMTP server configuration parameter" in the *Administration Guide: Performance*
- "contact_host - Location of contact list configuration parameter" in the *Administration Guide: Performance*

DAS Java virtual machine setup

Procedure:

The *jdk_path* configuration parameter specifies the directory under which the IBM Software Developer's Kit (SDK) for Java to be used for running DB2 administration server functions is installed. The environment variables used by the Java interpreter are computed from the value of this parameter.

The scheduler requires a Java virtual machine (JVM) in order to use the tools catalog database. It is necessary to have this setup before the scheduler can be successfully started.

There is no default value for this parameter when working with UNIX platforms. You should specify a value for this parameter when you install the IBM Software Developer's Kit (SDK) for Java.

The IBM Software Developer's Kit (SDK) for Java on Windows is installed under `%DB2PATH%\java\jdk` (which is the default value for this parameter on Windows platforms). This should already be specified to the DAS. You can verify the value for *jdk_path* using:

```
db2 get admin cfg
```

This command displays the values of the DB2 administration server configuration file where *jdk_path* is one of the configuration parameters. The parameter can be set, if necessary, using:

```
db2 update admin cfg using jdk_path 'C:\Program Files\IBM\SQLLIB'
```

This assumes that DB2 Universal Database™ (DB2 UDB) is installed under 'C:\Program Files\IBM\SQLLIB'.

The IBM Software Developer's Kit (SDK) for Java on AIX is installed under /usr/java130. The parameter can be set, if necessary, using:

```
db2 update admin cfg using jdk_path /usr/java130
```

Note: If you are creating or using a tools catalog against a 64-bit instance on one of the platforms that supports both 32- and 64-bit instances (AIX, Sun, or HP-UX) use the jdk_64_path configuration parameter instead of the jdk_path parameter. This configuration parameter specifies the directory under which the 64-bit version of the IBM Software Develop's Kit (SDK) for Java is installed.

Related reference:

- "GET ADMIN CONFIGURATION Command" in the *Command Reference*
- "UPDATE ADMIN CONFIGURATION Command" in the *Command Reference*
- "jdk_path - Software Developer's Kit for Java installation path DAS configuration parameter" in the *Administration Guide: Performance*

Security considerations for the DAS on Windows

You may need to change the user ID under which the DAS service runs on Windows®.

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. Before running this command, you must log on to a machine using an account or user ID that has local Administrator authority.

Note: Recall that passwords are case-sensitive. A mixture of upper and lower case is allowed which means that the case of the password becomes very important.

Note: On Windows, 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).

Related reference:

- "db2admin - DB2 Administration Server Command" in the *Command Reference*

Updating the DAS on UNIX

Procedure:

On UNIX operating systems, if DB2 is updated by installing a Program Temporary Fix (PTF) or a code patch, each DB2 administration server (DAS) and instance should be updated. To update the DAS, use the **dasupdt** 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 with superuser authority, usually as "root".

The command is used as follows:

```
dasupdt
```

There are also optional parameters for this command:

- –h or –?

  Displays a help menu for this command.
- –d

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

  Moves the DAS from a higher code level on one path to a lower code level installed on another path.

Note: On Windows, updating the DAS is part of the installation process. There are no user actions required.

Examples:

The DAS is running Version 8.1.2 code in the Version 8 install path. If FixPak 3 is installed in the Version 8 install path, the following command, invoked from the Version 8 install path, will update the DAS to FixPak 3:

```
dasupdt
```

The DAS is running Version 8.1.2 code in an alternate install path. If FixPak 1 is installed in another alternate install path, the following command, invoked from the FixPak 1 alternate install path, will update the DAS to FixPak 1, running from the FixPak 1 alternate install path:

```
dasupdt -D
```

Related concepts:

- "DB2 Administration Server" on page 39
- "Security considerations for the DAS on Windows" on page 50

Removing the DAS

Procedure:

To remove the DAS:

- On Windows operating systems:
  1. Log on to the machine using an account or user ID that has the correct authorization to remove a service.

2. Stop the DAS, using **db2admin stop**.
3. Backup (if needed) all the files in the `db2das00` subdirectory under the `sqllib` subdirectory.

   Note: This example assumes `db2das00` is the name of the DAS to be removed. It is possible to have a DAS with a name other than DB2DAS00 if a user has created a DB2 Universal Database™ (DB2 UDB) instance that has the name DB2DAS00. In this case, the DAS will be named DB2DAS01 (or, if that is taken, DB2DAS02 and so forth). You should look for the service with the "DB2DAS" prefix to identify the specific DAS from the list of several DAS that may exist. You can use the **db2admin** command without any options to list all DAS.
4. Drop the DAS, using **db2admin drop**.

- On UNIX operating systems:
  1. Login as a user with DASADM authority.
  2. Run the startup script using one of the following:

     ```
     . DASHOME/das/dasprofile    (for Bourne or Korn shell)
     source DASHOME/das/dascshrc (for C shell)
     ```

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

     ```
     db2admin stop
     ```
  4. Back up (if needed) all the files in the `das` subdirectory under the home directory of the DAS.
  5. Log off.
  6. Log in as `root` and remove the DAS using the **dasdrop** command as follows:

     ```
     dasdrop
     ```

     The **dasdrop** command is found in the `instance` subdirectory under the subdirectory specific to the installed DB2 UDB version and release.

     Note: The **dasdrop** command removes the `das` subdirectory under the home directory of the DB2 administration server (DAS).

Related reference:

- "db2admin - DB2 Administration Server Command" in the *Command Reference*
- "dasdrop - Remove a DB2 Administration Server Command" in the *Command Reference*

Setting up DAS with Enterprise Server Edition (ESE) systems

Procedure:

The following information shows the steps necessary to configure DB2 Universal Database™ (DB2 UDB) ESE servers (Linux, Solaris Operating Environment, Windows NT, Windows 2000, Windows Server 2003, 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 must create additional DAS on other machines to allow the Control Center or the Configuration Assistant access to other coordinator

nodes. The overhead of working as an administrative coordinator node can then be spread to more than one partition in an instance. The install program will create the DAS on all nodes that it is run on. Only if you do not use **db2setup** will you need to do this manually.

The directions given here are only applicable for a partitioned ESE environment. If you are only running on a single partition ESE system, then the directions given are not applicable to your environment.

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 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 UDB 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 ESE V8.1

install path:
install_path

TCP services file:
services

DB2 UDB Instance:

name: db2inst

owner ID:
db2inst

instance path:
instance_path

nodes: 3 nodes, db2nodes.cfg:

- 0 hostA 0 hostAswitch
- 1 hostA 1 hostAswitch
- 2 hostB 0 hostBswitch

DB name:
db2instDB

DAS:

name: db2as00

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 some sample supported ESE platforms:

Table 2. Example Pathnames for Supported ESE Platforms

Paths	DB2 UDB ESE for AIX	DB2 UDB ESE for Solaris Operating Environment	DB2 UDB ESE for Windows
install_path	/usr/opt/<v_r_ID>	/opt/IBM/db2/<v_r_ID>	C:\sqllib
instance_path	/home/db2inst/sqllib	/home/db2inst/sqllib	C:\profiles\db2inst
das_path	/home/db2as/das	/home/db2as/das	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 ESE for AIX in Version 8, the <v_r_ID> is db2_08_01.

When installing DB2 UDB ESE, 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 Configuration Assistant.

If DAS is installed on each physical machine, then each machine can act as a coordinator node. Each physical machine appears as a separate DB2SYSTEM in the Control Center or Configuration Assistant. If different clients use different systems to connect to a partitioned database server, then this will distribute the coordinator node functionality and help to balance incoming connections.

Related concepts:
- "DB2 Administration Server" on page 39
- "DAS configuration on Enterprise Server Edition (ESE) systems" on page 54

DAS configuration on Enterprise Server Edition (ESE) systems

The DAS is an administrative control point which performs certain tasks on behalf of the tools. There can be at most one DAS per physical machine. In the case of an ESE instance which consists of several machines, all of the machines must be running a DAS so that the Control Center can administer the ESE instance. This DAS (db2as) is represented by the system that is present in the Control Center navigator tree as the parent of the target DB2® Universal Database (DB2 UDB) instance (db2inst).

For example, db2inst consists of three nodes distributed across two physical machines or hosts. The minimum requirement can be fulfilled by running **db2as** on hostA and 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 the DAS on hostA regardless of the multiple logical nodes (MLN) configuration for that host.
2. There is one DAS required on each machine, or physical node, which must be created individually using the **dascrt** command. The DAS on each machine or physical node must be running so that the Task Center and the Control Center can work correctly. The ID db2as must exist on hostA and hostB. The home directory of the db2as ID must not be cross-mounted between the two systems. Alternatively, different user IDs can be used to create the DAS on hostA and hostB.

On DB2 Universal Database™ (DB2 UDB) Enterprise Server Edition for Windows®, if you are using the Configuration Assistant or the Control Center to automate connection configuration to a DB2 UDB 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 coordinator node before being routed to other database partition servers.

On DB2 Universal Database (DB2 UDB) Enterprise Server Edition for Windows, creating additional DB2 administration servers on other machines allows the Configuration Assistant or Control Center to configure other systems as coordinator nodes using DB2 Discovery.

When working on DB2 Universal Database (DB2 UDB) Enterprise Server Edition for Windows, the DB2 UDB Remote Command Service (**db2rcmd.exe**) automatically handles internode administrative communications.

The Control Center communicates with the DAS using the TCP service port 523. This port is reserved for exclusive use by DB2 UDB. Therefore, it is not necessary to insert new entries into TCP services file.

Related tasks:

- "Create a DB2 Administration Server" on page 41

Related reference:

- "db2admin - DB2 Administration Server Command" in the *Command Reference*

Discovery of administration servers, instances, and databases

To configure connections to a remote machine, there are two methods: using the discovery service that is built in to the Configuration Assistant; or, using an existing directory service such as Lightweight Directory Access Protocol (LDAP).

The discovery service is integrated with the Configuration Assistant and the DB2® administration server. To configure a connection to a remote machine, the user would logon to the client machine and run the Configuration Assistant (CA). The CA sends a broadcast signal to all the machines on the network. Any machine that has a DAS installed and configured for discovery will respond to the broadcast signal from the CA by sending back a package that contains all the instance and database information on that machine. The CA then uses the information in this

package to configure the client connectivity. Using the discovery method, catalog information for a remote server can be automatically generated in the local database and node directory.

The discovery method requires that you logon to every client machine and run the CA. If you have an environment where there are a large number of clients, this can be very difficult and time-consuming. An alternative, in this case, is to use a directory service like LDAP.

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 Universal Database™ (DB2 UDB) servers.

To have a system support Known Discovery, set the *discover* parameter in the DAS configuration file to `KNOWN`. To have the system support both Known and Search Discovery, set the *discover* parameter in the DAS configuration file to `SEARCH` (this is the default). To prevent discovery of a system, and all of its instances and databases, set this parameter to `DISABLE`. Setting the *discover* parameter to `DISABLE` in the DAS configuration file, prevents discovery of the system.

Note: The TCP/IP host name returned to a client by Search Discovery is the same host name that is returned by your DB2 UDB 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 UDB 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**

  KNOWN discovery is used by the Configuration Assistant and Control Center to retrieve instance and database information associated with systems that are already known to your local system. New systems can be added using the **Add Systems** functionality provided in the tools. When the *discover* parameter is set to `KNOWN`, you will not be able to search the network.
- **SEARCH**

  Enables all of the facilities of Known Discovery, and enables local network searching. This means that any searching is limited to the local network.

  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).

Related concepts:

- "Lightweight Directory Access Protocol (LDAP) Directory Service" on page 68

Related tasks:

- "Hiding server instances and databases from discovery" on page 57
- "Setting discovery parameters" on page 57

Hiding server instances and databases from discovery

Procedure:

You may have multiple instances, and multiple databases within these instances, on a server system. 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.

Note: If you want an instance to be discovered, discover must also be set to `KNOWN` or `SEARCH` in the DAS configuration file. If you want a database to be discovered, the *discover_inst* parameter must also be enabled in the server instance.

Related reference:

- "discover_inst - Discover server instance configuration parameter" in the *Administration Guide: Performance*
- "discover_db - Discover database configuration parameter" in the *Administration Guide: Performance*

Setting discovery parameters

Procedure:

The *discover* parameter is set in the DAS configuration file on the server system, and in the database manager configuration file on the client. Use the Configuration Assistant or Control Center to set the database manager configuration parameters: *discover*, *discover_inst*, *discover_db*. Set the parameters as follows:

- On the DAS:

  Update the *discover* parameter (as an example) in the DAS configuration file using the command process:

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

  The DAS *discover* configuration parameter is configurable online which means that it is not necessary for you to stop and restart the DAS for the change to take effect.

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

- By working with the Configuration Assistant:

  Start the Configuration Assistant by entering **db2ca** from the command line (on all platforms) or from the Start menu (on Windows): Click **Start —> Programs —> IBM DB2 —> Set-up Tools —> Configuration Assistant**.

  To use the Configuration Assistant to set the database manager configuration parameters:

1. Click **Configure —> DBM Configuration**.
2. Click the keyword that you want to modify.
3. In the value column, click a value for the keyword that you want to modify, and click **OK**.
4. Click **OK** again, a message displays. Click **Close**.

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

You can also use the Configuration Assistant to update configuration parameters.

Related reference:

- "discover_inst - Discover server instance configuration parameter" in the *Administration Guide: Performance*
- "discover_db - Discover database configuration parameter" in the *Administration Guide: Performance*
- "UPDATE ADMIN CONFIGURATION Command" in the *Command Reference*
- "discover - DAS discovery mode configuration parameter" in the *Administration Guide: Performance*

Setting up the DAS to use the Configuration Assistant and the Control Center

Prerequisites:

You must configure **discover** to retrieve information about systems on your network.

Restrictions:

A DAS must reside on each physical partition. When a DAS is created on the partition, the DB2SYSTEM name is configured to the TCP/IP hostname and the **discover** setting is defaulted to SEARCH.

Procedure:

DB2 Discovery is a feature that is used by the Configuration Assistant and Control Center. Configuring for this feature may require that you update the DB2 administration server (DAS) configuration and an instance's database manager configuration to ensure that DB2 Discovery retrieves the correct information.

When a client issues a discovery request from the Configuration Assistant, or Control Center, each DAS with discovery enabled will respond. In a partitioned database environment, each physical partition will respond as a separate DB2SYSTEM name. The actual instances that can be administered depend on the instances known by that physical partition. Since instances can span multiple

partitions, the same instance can potentially be administered through different system names. You can use this capability to help you balance the load on the server instance. For example, if an instance "A" is available through system "S1" and system "S2", then some users could catalog a database using "S1" and some could catalog the same database using "S2". Each user could connect to the server using a different coordinator database partition.

Related reference:

- "db2ilist - List Instances Command" in the *Command Reference*
- "db2ncrt - Add Database Partition Server to an Instance Command" in the *Command Reference*
- "discover - DAS discovery mode configuration parameter" in the *Administration Guide: Performance*

Update the DAS configuration for discovery

Restrictions:

A DAS must reside on each physical partition. When a DAS is created on the partition, the DB2SYSTEM name is configured to the TCP/IP hostname and the *discover* setting is defaulted to SEARCH.

Procedure:

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.

When updating a DAS configuration, and you want to be able to select a coordinator node from a list of DB2 Universal Database™ (DB2 UDB) systems, set *discover*=SEARCH (which is the default) in each DB2 administration server's configuration file.

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

Related reference:

- "Miscellaneous variables" in the *Administration Guide: Performance*

DB2 administration server first failure data capture

First failure data capture (FFDC) is a general term applied to the set of diagnostic information the DB2® administration server captures automatically when errors occur. This information reduces the need to reproduce errors to get diagnostic information. The diagnostic information is contained in a single location.

The information captured by the DB2 administration server FFDC includes:

- Administration notification logs.

When an event occurs, the DB2 administration server writes information to the DB2 administration server log file, db2dasdiag.log.

- Dump files.

  For some error conditions, extra information is logged in external binary dump files named after the failing process ID. These files are intended for use by DB2 Universal Database™ (DB2 UDB) Customer Support.

- Trap files.

  The DB2 administration server generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception. Trap files contain a function flow of the last steps that were run before a problem occurred.

DB2 administration server first failure data capture information location.

By default, the DB2 administration server FFDC information is placed in the following locations:

- On Windows® systems:

  If the DB2INSTPROF environment variable is not set:

  ```
  db2path\DB2DAS00\dump
  ```

  where db2path is the path referenced in the DB2PATH environment variable, and DB2DAS00 is the name of the DAS service. The DAS name can be obtained by typing the **db2admin** command without any arguments.

  If the DB2INSTPROF environment variable is set:

  ```
  x:\db2instprof\DB2DAS00\dump
  ```

  where x: is the drive referenced in the DB2PATH environment variable, db2instprof is the instance profile directory, and DB2DAS00 is the name of the DAS service.

- On UNIX®-based systems:

  ```
  $DASHOME/das/dump
  ```

  where $DASHOME is the home directory of the DAS user.

Note: You should clean out the dump directory periodically to keep it from becoming too large.

Interpreting the DB2 administration server log.

The format of the DB2 administration server log file (db2dasdiag.log) is similar to the format of the DB2 FFDC log file db2diag.log. Refer to the section on interpreting the administration logs in the troubleshooting topics for information about how to interpret the db2dasdiag.log file.

Related concepts:

- "DB2 Administration Server" on page 39

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.

Creating a database

Prerequisites:

You should have spent sufficient time designing the contents, layout, potential growth, and use of your database before you create it.

Procedure:

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 —> Standard** or **Create —> With Automatic Maintenance** 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"
```

When creating a database, you can also request to use the Configuration Advisor to assist the configuration of the database instead of accepting the default values for all of the configuration parameters. You can do this by using the AUTOCONFIGURE option on the **CREATE DATABASE** command:

```
CREATE DATABASE <database name>
   AUTOCONFIGURE
```

There are several options on the AUTOCONFIGURE clause. You cannot use the AUTOCONFIGURE clause when creating a database in a partitioned environment.

At the same time a database is created, a detailed deadlocks event monitor is also created. As with any monitor, there is some overhead associated with this event monitor. If you do not want the detailed deadlocks event monitor, then the event monitor can be dropped using the command:

```
DROP EVENT MONITOR db2detaildeadlock
```

To limit the amount of disk space that this event monitor consumes, the event monitor deactivates, and a message is written to the administration notification log, once it has reached its maximum number of output files. Removing output files that are no longer needed allows the event monitor to activate again on the next database activation.

You have the ability to create a database in a different, possibly remote, database manager instance. In this type of environment you have the ability to perform instance-level administration against an instance other than your default instance, including remote instances.

Related concepts:

- "What to record in a database" in the *Administration Guide: Planning*
- "Multiple instances of the database manager" on page 5
- "Database authorities" on page 225
- "Additional database design considerations" in the *Administration Guide: Planning*

Related reference:

- "CREATE DATABASE Command" in the *Command Reference*

Definition of initial database partition groups

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 DBPARTITIONNUM** and **DROP DBPARTITIONNUM VERIFY** commands.

Three database partition groups 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.

Related concepts:

- "Database partition groups" in the *Administration Guide: Planning*

Related reference:

- "ADD DBPARTITIONNUM Command" in the *Command Reference*
- "DROP DBPARTITIONNUM VERIFY Command" in the *Command Reference*

Defining initial table spaces

When a database is created, three table spaces are defined:

- SYSCATSPACE for the system catalog tables
- 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. The extent size for these table spaces is set to the default.

Prerequisites:

The database must be created and you must have the authority to create table spaces.

Procedure:

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 —> Standard** or **Create —> With Automatic Maintenance** 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 Windows:

```
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.

Note: When working in a partitioned database environment, you cannot create or assign containers to specific partitions. First, you must create the database with default user and temporary table spaces. Then you should use the CREATE TABLESPACE statement to create the required table spaces. Finally, you can drop the default table spaces.

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 statement.

Related concepts:

- "Definition of system catalog tables" on page 65
- "Table space design" in the *Administration Guide: Planning*

Related tasks:

- "Creating a table space" on page 73

Related reference:

- "CREATE DATABASE Command" in the *Command Reference*

Creating a buffer pool

You can create new buffer pools for use by the database manager. Buffer pools improve database system performance immediately.

The page sizes specified for your table spaces should determine the page sizes that you choose for your buffer pools. The choice of page size used for a buffer pool is important because you cannot alter the page size after you create a buffer pool.

Prerequisites:

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

Before you create a new buffer pool, resolve the following questions:

- What buffer pool name to use
- Whether the buffer pool is to be created immediately or created following the next time that the database is deactivated and reactivated
- Whether you want to associate the buffer pool with a subset of all database partitions that make up the database
- What values you want to associate with the parameters controlling the size of the buffer pool, including the page size and the total size of the buffer pool based on the number of pages
- Whether you want to use extended storage, block-based support, or neither.

Procedure:

To create a new buffer pool:

1. `SELECT BPNAME FROM SYSCAT.BUFFERPOOLS` to get the list of buffer pool names that already exist in the database.

2. Choose a buffer pool name that is not currently found in the result list. The name must not begin with the characters "SYS" or "IBM."
3. Determine the characteristics of the buffer pool you are going to create.
4. Ensure that you have the correct authorization ID to run the CREATE BUFFERPOOL statement.
5. Run the CREATE BUFFERPOOL statement.

Related tasks:

- "Altering a buffer pool" on page 162

Related reference:

- "CREATE BUFFERPOOL statement" in the *SQL Reference, Volume 2*

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 routines (functions and procedures) in the schemas SYSIBM, SYSFUN, and SYSPROC.
- A set of read-only views for the system catalog tables is created in the SYSCAT schema.
- 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.

After your database has been created, you may wish to limit the access to the system catalog views.

Related concepts:

- "User-defined functions" in the *SQL Reference, Volume 1*
- "Catalog views" in the *SQL Reference, Volume 1*
- "Functions overview" in the *SQL Reference, Volume 1*

Related tasks:

- "Securing the system catalog view" on page 247

Related reference:

- "Functions" in the *SQL Reference, Volume 1*

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 (or "drive" for Windows® 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.

Related reference:

- "CREATE DATABASE Command" in the *Command Reference*

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 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 current database manager instance
- Other system information.

On UNIX® platforms and 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.

Related tasks:

- "Enabling data partitioning in a database" on page 11

- "Cataloging a database" on page 71

Related reference:

- "CREATE DATABASE Command" in the *Command Reference*

Identify an alternate server for a database

Whenever a server crashes, each client that is connected to that server receives a communications error which terminates the connection resulting in an application error. In cases where availability is important, you should have implemented either a redundant set up or the ability to fail the server over to a standby node. In either case, the DB2 Universal Database™ (DB2 UDB) client code attempts to re-establish the connection to the original server which may be running on a failover node (the IP address fails over as well), or to a new server.

Procedure:

To define a new or alternate server, use the UPDATE ALTERNATE SERVER FOR DATABASE command. This command updates the alternate server information for a database alias in the system database directory.

Related concepts:

- "Automatic client reroute implementation" on page 70
- "Automatic client reroute description and setup" on page 299

Viewing the local or system database directory files

You would like to see some of the information associated with the databases that you have on your system.

Prerequisites:

Before viewing either the local or system database directory files, you must have previously created an instance and a database.

Procedure:

To see the contents of the local database directory file, issue the following command, where `<location>` specifies the location of the database:

```
LIST DATABASE DIRECTORY ON <location>
```

To see the contents of the system database directory file, issue the **LIST DATABASE DIRECTORY** command *without* specifying the location of the database directory file.

Related reference:

- "LIST DATABASE DIRECTORY Command" in the *Command Reference*

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.

Related reference:

- "CATALOG TCPIP NODE Command" in the *Command Reference*
- "LIST NODE DIRECTORY Command" in the *Command Reference*
- "CATALOG NETBIOS NODE Command" in the *Command Reference*
- "CATALOG LOCAL NODE Command" in the *Command Reference*
- "CATALOG NAMED PIPE NODE Command" in the *Command Reference*

Lightweight Directory Access Protocol (LDAP) Directory Service

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. Clients and servers would use the directory service to find out how to access other resources. Information about these other resources in the distributed environment must be entered into the directory service repository.

Lightweight Directory Access Protocol (LDAP) is an industry standard access method to directory services. 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.

As an administrator of a DB2® UDB system, you can establish and maintain a directory service. The Configuration Assistant or Control Center can assist in the maintenance of this directory service. The directory service is made available to DB2 UDB through Lightweight Directory Access Protocol (LDAP) directory services. To use LDAP directory services, there must first exist an LDAP server that is supported by DB2 so that directory information can be stored there.

Note: When running in a Windows® 2000 domain environment, an LDAP server is already available because it is integrated with the Windows 2000 Active Directory. As a result, every machine running Windows 2000 can use LDAP.

An LDAP directory is helpful in an enterprise environment where it is difficult to update local directory catalogs on each client machine because of the large number of clients. When this is your situation, it is recommended you store directory entries in an LDAP server so that maintaining catalog entries is done in one place: on the LDAP server. The cost of purchasing and maintaining an LDAP server can be significant and so should only be considered when there are sufficient numbers of clients to offset the cost.

Related concepts:

- "Discovery of administration servers, instances, and databases" on page 55
- "Introduction to Lightweight Directory Access Protocol (LDAP)" on page 305

Creating database partition groups (nodegroups)

You create a database partition group with the CREATE DATABASE PARTITION GROUP statement. This statement specifies the set of database partitions on which the table space containers and table data are to reside. This statement also:

- Creates a partitioning map for the database partition group.
- Generates a partitioning map ID.
- Inserts records into the following catalog tables:
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.DBPARTITIONGROUPDEF

Prerequisites:

The machines and systems must be available and capable of handling a partitioned database environment. You have purchased and installed DB2 Universal Database Enterprise - Server Edition. The database must exist.

Procedure:

To create a database partition group using the Control Center:

1. Expand the object tree until you see the **Database partition groups** folder.
2. Right-click the **Database partition groups** folder, and select **Create** from the pop-up menu.
3. On the Create Database partition groups window, complete the information, use the arrows to move nodes from the **Available nodes** box to the **Selected database partitions** box, and click **Ok**.

To create a database partition group using the command line, enter:

```
CREATE DATABASE PARTITION GROUP <name> ON PARTITIONS (<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 database partition group of two database partitions (1 and 2) in a database consisting of at least three (0 to 2) database partitions:

```
CREATE DATABASE PARTITION GROUP mixng12 ON PARTITIONS (1,2)
```

The **CREATE DATABASE** command or sqlecrea() API also create the default system database partition groups, IBMDEFAULTGROUP, IBMCATGROUP, and IBMTEMPGROUP.

Related concepts:

- "Database partition groups" in the *Administration Guide: Planning*
- "Partitioning maps" in the *Administration Guide: Planning*

Related reference:

- "CREATE DATABASE PARTITION GROUP statement" in the *SQL Reference, Volume 2*
- "sqlecrea - Create Database" in the *Administrative API Reference*
- "CREATE DATABASE Command" in the *Command Reference*

Definition of the 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.

Related concepts:

- "Understanding recovery logs" in the *Data Recovery and High Availability Guide and Reference*

Automatic client reroute implementation

When a DB2® Universal Database (DB2 UDB) client application suffers from a loss of communication with a DB2 UDB server, you would like the client to be able to recover from such a loss without any intervention by you or another administrator. DB2 UDB supports the recovery of client communication to a DB2 UDB server. What needs to occur before the communication failure is the establishment of an alternative location that the client connection knows.

The UPDATE ALTERNATE SERVER FOR DATABASE command is used to define the alternate server location on a particular database. The alternate hostname and port number is given as part of the command. The location is stored in the system database directory file at the server.

After you have specified the alternate server location on a particular database at the server instance, the alternate server location information is returned to the client as part of the connection process. If communication between the client and the server is lost for any reason, the DB2 UDB client coded will attempt to re-establish the connection by using the alternate server information. The DB2 UDB client will attempt to re-connect with the original server and the alternate server, alternating the attempts between the two servers. The timing of these attempts varies from very rapid attempts to begin with gradual lengthening of the intervals between the attempts.

Once a connection is successful, the SQLCODE -30108 is returned to indicate that a database connection has been re-established following the communication failure. The hostname/IP address and service name/port number are returned. The client code only returns the error for the original communications failure to the application if the re-establishment of the client communications is not possible to either the original or alternative server.

Related concepts:

- "Automatic client reroute description and setup" on page 299
- "Automatic client reroute limitations" on page 300

Related reference:

- "Automatic client reroute examples" on page 301

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.

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.

Related tasks:

- "Creating a database" on page 61

Related reference:

- "BIND Command" in the *Command Reference*

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.

Prerequisites:

Although databases are cataloged automatically when a database is created, you still may have a need to catalog the database. When you do so, the database must exist.

Procedure:

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 `INSTNC_C`. The instance `instnc_c` must already be cataloged as a local node before attempting this command.

```
CATALOG DATABASE b as b_on_ic AT NODE instnc_c
```

Note: The **CATALOG DATABASE** command is also used on client nodes to catalog databases that reside on database server machines.

Note: By default directory files, including the database directory, are cached in memory using the "Directory Cache Support (*dir_cache*)" configuration parameter. 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 a partitioned database, a cache for directory files is created on each database partition.

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.

Related tasks:

- "Updating the directories with information about remote database server machines" on page 72

Related reference:

- "dir_cache - Directory cache support configuration parameter" in the *Administration Guide: Performance*
- "CATALOG DATABASE Command" in the *Command Reference*
- "TERMINATE Command" in the *Command Reference*
- "UNCATALOG DATABASE Command" in the *Command Reference*

Updating the directories with information about remote database server machines

Procedure:

You can use the Add Database Wizard of the Configuration Assistant (CA) interpreter to create catalog entries. If you have the DB2 Application Development Client, you can also create an application program to catalog entries.

Note: To catalog a database, you must have SYSADM or SYSCTRL authority; or, you must have the *catalog_noauth* configuration parameter set to YES.

To update the directories using the command line processor, do the following:

1. Use one of the following commands to update the node directory:
   - For a node having an APPC connection:

```
db2 CATALOG APPC NODE <nodename>
  REMOTE <symbolic_destination_name> SECURITY <security_type>
```

For example:

```
db2 CATALOG APPC NODE DB2NODE REMOTE DB2CPIC SECURITY PROGRAM
```

- For a DB2 Universal Database for z/OS and OS/390 Version 5.1 (or later) or a DB2 Universal Database for AS/400 Version 4.2 (or later) database having a TCP/IP connection:

```
db2 CATALOG TCPIP NODE <nodename>
   REMOTE <hostname> or <IP address>
   SERVER <service_name> or <port_number>
   SECURITY <security_type>
```

For example:

```
db2 CATALOG TCPIP NODE MVSIPNOD REMOTE MVSHOST SERVER DB2INSTC
```

The default port used for TCP/IP connections on DB2 for OS/390 and z/OS is 446.

2. If you work with DB2 Connect, you will have to consider updating the DCS directory using the CATALOG DCS DATABASE command.

If you have remote clients, you must also update directories on each remote client.

Related concepts:

- "System database directory values" in the *DB2 Connect User's Guide*
- "DCS directory values" in the *DB2 Connect User's Guide*

Related reference:

- "CATALOG DATABASE Command" in the *Command Reference*
- "CATALOG DCS DATABASE Command" in the *Command Reference*
- "CATALOG APPC NODE Command" in the *Command Reference*
- "CATALOG TCPIP NODE Command" in the *Command Reference*
- "CATALOG NETBIOS NODE Command" in the *Command Reference*
- "CATALOG APPN NODE Command" in the *Command Reference*

Creating a table space

Table spaces establish the relationship between the physical storage devices used by your database system and the logical containers or tables used to store data.

Prerequisites:

You must know the device or file names of the containers that you will reference when creating your table spaces. In addition, you must know the space associated with each device or file name that you will allocate to the table space.

Procedure:

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.

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 Windows 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 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 previous two examples, explicit names are provided for the containers. However, if you specify relative container names, the container is created in the subdirectory created for the database.

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 database partition group. The default database partition group IBMDEFAULTGROUP is used when the following parameter is not specified in the statement:

```
IN database_partition_group_name
```

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 12.67
   TRANSFERRATE 0.18
```

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 database partition group called ODDGROUP in a UNIX partitioned database. ODDGROUP must be previously created with a CREATE DATABASE PARTITION GROUP statement. In this case, the ODDGROUP database partition group 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 IN ODDGROUP
   MANAGED BY DATABASE
   USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000)
          ON DBPARTITIONNUM 1
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000)
          ON DBPARTITIONNUM 3
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000)
          ON DBPARTITIONNUM 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. The current defaults are:

- OVERHEAD 12.67 ms
- TRANSFERRATE 0.18 ms

DB2 UDB can greatly improve the performance of sequential I/O using the sequential prefetch facility, which uses parallel I/O.

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.

You can use the ALTER TABLESPACE SQL statement to add, drop, or resize containers to a DMS table space and modify the PREFETCHSIZE, OVERHEAD, and TRANSFERRATE settings for a table space. You should commit the transaction issuing the table space statement 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. You should use the following equation to set your prefetch size manually when creating a table space:

```
prefetch size = (number of containers) X (number of physical spindles per
                    container) X extent size
```

You should also consider letting DB2 UDB automatically determine the prefetch size.

Related concepts:

- "Table space design" in the *Administration Guide: Planning*
- "System managed space" in the *Administration Guide: Planning*
- "Database managed space" in the *Administration Guide: Planning*
- "Sequential prefetching" in the *Administration Guide: Performance*

Related tasks:

- "Enabling large page support in a 64-bit environment (AIX)" in the *Administration Guide: Planning*

Related reference:

- "ALTER TABLESPACE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLESPACE statement" in the *SQL Reference, Volume 2*

Creating specific types of table spaces

There are different types of tables spaces that are used by the database manager and for use by applications and users.

Creating a system temporary table space

Although a system temporary table space is created by default when you create a database, you may want to allocate a separate table space to work on system sort tasks.

Prerequisites:

The containers to be associated with the system temporary table space must exist.

Restrictions:

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.

Procedure:

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".

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')
```

You should have at least one table space of each pagesize.

The only database partition group that can be specified when creating a system temporary table space is IBMTEMPGROUP.

Related tasks:

- "Creating a user temporary table space" on page 77

Related reference:

- "CREATE TABLESPACE statement" in the *SQL Reference, Volume 2*

Creating a user temporary table space

User temporary table spaces are not created by default when a database is created. As part of the work your application programs may be doing with the data in the database, you may need to use temporary tables. If so, you will need to create a user temporary table.

Procedure:

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 database partition group other than IBMTEMPGROUP. The default database partition group 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.

Related tasks:

- "Creating a user-defined temporary table" on page 96

Related reference:

- "CREATE TABLESPACE statement" in the *SQL Reference, Volume 2*
- "DECLARE GLOBAL TEMPORARY TABLE statement" in the *SQL Reference, Volume 2*

Creating table spaces in database partition groups

By placing a table space in a multiple-partition database partition group, all of the tables within the table space are divided or partitioned across each partition in the database partition group. The table space is created into a database partition group. Once in a database partition group, the table space must remain there; it

cannot be changed to another database partition group. The CREATE TABLESPACE statement is used to associate a table space with a database partition group.

Related reference:

- "CREATE TABLESPACE statement" in the *SQL Reference, Volume 2*

Specifying raw I/O

When working with containers to store data, DB2 Universal Database supports direct disk access (raw I/O). This type of support allows you to attach a direct disk access (raw) device to any DB2 Universal Database system. (The only exception is the Windows 9x operating system.)

Prerequisites:

You must know the device or file names of the containers you are going to reference when creating your table spaces. You must know the amount of space associated with each device or file name that is to be allocated to the table space.

You will need the correct permissions to read and write to the container.

Procedure:

The following list demonstrates the physical and logical methods for identifying a direct disk access 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:

  ```
  \\.\PhysicalDrive5
  ```

- On Windows, to specify a logical drive (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. To overcome the limitation imposed by using a letter to identify the drive, you can use a globally unique identifier (GUID) with the logical drive.

- **Note:** You must have Windows NT Version 4.0 with Service Pack 3 or later installed to be able to write to a device.

- On UNIX-based platforms, a logical volume can appear to users and applications as a single, contiguous, and extensible disk volume. Although it appears this way, it can reside on noncontiguous physical partitions or even on more than one physical volume. The logical volume must also be contained within a single volume group. There is a limit of 256 logical volumes per volume group. There is a limit of 32 physical volumes per volume group. You can create additional logical volumes using the **mklv** command. This command

allows you to specify the name of the logical volume and to define its characteristics, including the number and location of logical partitions to allocate for it.

After you create a logical volume, you can change its name and characteristics with the **chlv** command, and you can increase the number of logical partitions allocated to it with the **extendlv** command. The default maximum size for a logical volume at creation is 512 logical partitions, unless specified to be larger. The **chlv** command is used to override this limitation.

Within AIX, the set of operating system commands, library subroutines, and other tools that allow you to establish and control logical volume storage is called the Logical Volume Manager (LVM). The LVM controls disk resources by mapping data between a simpler and flexible logical view of storage space and the actual physical disks.

For more information on the **mklv** and other logical volume commands, and the LVM, refer to *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*.

For Windows 2000 and above, there is a new method for specifying DMS raw table space containers. Volumes (that is, basic disk partitions or dynamic volumes) are assigned a globally unique identifier (GUID) when they are created. The GUID can be used as a device identifier when specifying the containers in a table space definiton. The GUIDs are unique across systems which means that in a multiple partitioned database configuration, GUIDs are different for each partition even if the disk partition defintions are the same.

A tool called *db2listvolumes.exe* is available (only on Windows operating systems) to make it easy to display the GUIDs for all the disk volumes defined on a Windows system. This tool creates two files in the current directory where the tool is run. One file, called `volumes.xml`, contains information about each disk volume encoded in XML for easy viewing on any XML-enabled browser. The second file, called `tablespace.ddl`, contains the required syntax for specifying table space containers. This file must be updated to fill in the remaining information needed for a table space definition. The *db2listvolumes* tool does not require any command line arguments.

Related tasks:

• "Setting up raw I/O on Linux" on page 79

Setting up raw I/O on Linux

When working with containers to store data, DB2 Universal Database supports direct disk access (raw I/O). This type of support allows you to attach a direct disk access (raw) device to any DB2 Universal Database system. There is specific information while working in a Linux environment.

Prerequisites:

You must know the device or file names of the containers you are going to reference when creating your table spaces. You must know the space associated with each device or file name that is to be allocated to the table space.

Before you set up raw I/O on Linux, you require the following:

- One or more free IDE or SCSI disk partitions

- 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

Note: Of the distributions currently supporting raw I/O, the naming of raw device nodes is different:

Table 3. Linux distributions supporting raw I/O.

Distribution	Raw device nodes	Raw device controller
RedHat or TurboLinux	/dev/raw/raw1 to 255	/dev/rawctl
SuSE	/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.

Procedure:

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.

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
```

Table 4. Linux raw I/O calculations.

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 appropriate read permissions on the raw device controller and the disk partition. Set appropriate read and write permissions on the raw device.

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.

Related tasks:

- "Specifying raw I/O" on page 78

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.

Prerequisites:

The database tables and other related objects that are to be grouped together must exist.

Restrictions:

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.

Unqualified 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.

The new schema name cannot already exist in the system catalogs and it cannot begin with ″SYS″.

Procedure:

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.

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 within a database.
2. Right-click the **Schema** folder, and click **Create**.
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
```

Related concepts:

- "Grouping objects by schema" on page 7
- "Implicit schema authority (IMPLICIT_SCHEMA) considerations" on page 227
- "Schema privileges" on page 227

Related tasks:

- "Setting a schema" on page 82

Related reference:

- "CREATE SCHEMA statement" in the *SQL Reference, Volume 2*

Details on the creation of schemas

Schemas are used to organize object ownership within the database.

Setting a schema

Once you have several schemas in existence, you may want to designate one as the default for schema for use by unqualified object references in dynamic SQL statements issued from within a specific DB2 connection.

Procedure:

Establishing a default schema 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 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.

Related concepts:

- "Schemas" in the *SQL Reference, Volume 1*

Related reference:

- "SET SCHEMA statement" in the *SQL Reference, Volume 2*
- "Reserved schema names and reserved words" in the *SQL Reference, Volume 1*
- "CURRENT SCHEMA special register" in the *SQL Reference, Volume 1*

Chapter 4. Creating tables and other related table objects

This chapter describes how to create tables with specific characteristics when implementing your database design.

Creating and populating a table

Tables are the main repository of data within databases. Creating the tables and entering data to fill the tables will occur when you are creating an new database.

Prerequisites:

You must take the time to design and organize the tables that will hold your data.

Procedure:

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 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 database partition group 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
- The LOAD or IMPORT commands
- The autoloader utility if working in a partitioned database environment

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.

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.

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.

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 click **Create**.
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.

To declare a global temporary table, use the DECLARE GLOBAL TEMPORARY TABLE statement.

You can also create a table that is defined based on the result of a query. This type of table is called a *materialized query table*.

Related concepts:

- "Import Overview" in the *Data Movement Utilities Guide and Reference*
- "Load Overview" in the *Data Movement Utilities Guide and Reference*
- "Moving data across platforms - file format considerations" in the *Data Movement Utilities Guide and Reference*
- "User-defined type (UDT)" on page 115

Related tasks:

- "Creating a materialized query table" on page 121

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- "INSERT statement" in the *SQL Reference, Volume 2*
- "DECLARE GLOBAL TEMPORARY TABLE statement" in the *SQL Reference, Volume 2*
- "IMPORT Command" in the *Command Reference*
- "LOAD Command" in the *Command Reference*

Details on creating and populating a table

Tables contain all of your data. There are many things you should consider when creating the tables and placing data within them.

Introduction to space compression for tables

There are two ways in which tables can occupy less space when stored on disk:

- If the column value is NULL, do not set aside the defined, fixed amount of space.
- If the column value can be easily known or determined (like default values) and if the value is available to the database manager during record formatting and column extraction.

DB2® Universal Database (DB2 UDB) has an optional record format that allows for this type of space savings. Space saving can take place at the table level as well as at the column level.

Related concepts:

- "Space requirements for database objects" in the *Administration Guide: Planning*
- "Space compression for new tables" on page 87
- "Space compression for existing tables" on page 165

Space compression for new tables

When creating a table, you can use the optional VALUE COMPRESSION clause to specify that the table is using the space saving row format at the table level and possibly at the column level.

When VALUE COMPRESSION is used, NULLs and zero-length data that has been assigned to defined variable-length data types (VARCHAR, VARGRAPHICS,

LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, and DBCLOB) will not be stored on disk. Only overhead values associated with these data types will take up disk space.

If VALUE COMPRESSION is used then the optional COMPRESS SYSTEM DEFAULT option can also be used to further reduce disk space usage. Minimal disk space is used if the inserted or updated value is equal to the system default value for the data type of the column. The default value will not be stored on disk. Data types that support COMPRESS SYSTEM DEFAULT include all numerical type columns, fixed-length character, and fixed-length graphic string data types. This means that zeros and blanks can be compressed.

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

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 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.

Related concepts:

- "Space requirements for large object data" in the *Administration Guide: Planning*

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- "Large objects (LOBs)" in the *SQL Reference, Volume 1*

Defining constraints

This section discusses how to define constraints:

- "Defining a unique constraint"
- "Defining referential constraints" on page 91
- "Defining a table check constraint" on page 94
- "Defining an informational constraint" on page 95.

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.

Restrictions:

A unique constraint may not be defined on a subtable.

There can be only one primary key per table.

Procedure:

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.

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). 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.

Related concepts:

- "Keys" in the *SQL Reference, Volume 1*
- "Constraints" in the *SQL Reference, Volume 1*

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Defining referential constraints

Referential integrity is imposed by adding referential constraints to table and column definitions. Once referential constraints are defined to the database manager, changes to the data within the tables and columns is checked against the defined constraint. Completion of the requested action depends on the result of the constraint checking.

Procedure:

Referential constraints are established with the FOREIGN KEY clause, and the REFERENCES clause in the CREATE TABLE or ALTER TABLE statements. There are effects from a referential constraint on a typed table or to a parent table that is a typed table that you should consider before creating a referential constraint.

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.

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. You could also use the CREATE SCHEMA statement to create both the EMPLOYEE and DEPARTMENT tables at the same time.

Related concepts:

- "Foreign key clause" on page 92
- "References clause" on page 93

Related tasks:

- "Adding foreign keys" on page 173

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE SCHEMA statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

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.

Related tasks:

- "Defining a unique constraint" on page 89
- "Defining referential constraints" on page 91

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

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.

Related concepts:

- "Foreign key clause" on page 92

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

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 statement:

```
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.

Related concepts:

- "Import Overview" in the *Data Movement Utilities Guide and Reference*
- "Load Overview" in the *Data Movement Utilities Guide and Reference*
- "Checking for integrity violations" in the *Data Movement Utilities Guide and Reference*

Related reference:

- "SET INTEGRITY statement" in the *SQL Reference, Volume 2*

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. Once table check constraints are defined to the database manager, an insert or update to the data within the tables is checked against the defined constraint. Completion of the requested action depends on the result of the constraint checking.

Procedure:

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.

Related concepts:

- "Constraints" in the *SQL Reference, Volume 1*

Related tasks:

- "Adding a table check constraint" on page 174

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- "ALTER SERVER statement" in the *SQL Reference, Volume 2*

Defining an informational constraint

An *informational constraint* is a rule that can be used by the SQL compiler but is not enforced by the database manager. The SQL compiler includes a rewrite query stage which transforms SQL statements into forms that can be optimized and improve the access path to the required data. The purpose of the constraint is not to have additional verification of data by the database manager, rather it is to improve query performance.

Procedure:

You define informational constraints using the CREATE TABLE or ALTER TABLE statements. Within those statements you add referential integrity or check constraints. You then associate constraint attributes to them specifying whether you want the database manager to enforce the constraint or not; and, whether you want the constraint to be used for query optimization or not.

Related concepts:

- "Constraints" in the *SQL Reference, Volume 1*
- "The SQL compiler process" in the *Administration Guide: Performance*
- "Query rewriting methods and examples" in the *Administration Guide: Performance*

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

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.

Procedure:

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.

Related tasks:

- "Defining a generated column on an existing table" on page 178

Related reference:

- "CREATE INDEX statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- "SELECT statement" in the *SQL Reference, Volume 2*

Creating a user-defined temporary table

A user-defined temporary table is needed by applications you are writing to work with data in the database. Results from manipulation of the data need to be stored temporarily in a table.

Prerequisites:

A user temporary table space must exist before creating a user-defined temporary table.

Restrictions:

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.

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

Procedure:

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.

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.

When a ROLLBACK or ROLLBACK TO SAVEPOINT is specified when creating this table, either you can specify to delete all the rows in the table (DELETE ROWS, which is the default), or you can specify that the rows of the table are to be preserved (PRESERVE ROWS).

Related tasks:

- "Creating a user temporary table space" on page 77

Related reference:

- "ROLLBACK statement" in the *SQL Reference, Volume 2*
- "SAVEPOINT statement" in the *SQL Reference, Volume 2*

- "DECLARE GLOBAL TEMPORARY TABLE statement" in the *SQL Reference, Volume 2*

Defining an identity column on a new table

An *identity column* provides a way for DB2 to automatically generate a 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. To guarantee a unique numeric value for each row that is added to a table, you should define a unique index on the identity column or declare it a primary key.

Restrictions:

Once created, you cannot alter the table description to include an identity column.

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 if the uniqueness of the values in the identity column is being enforced by a primary-key or a unique index that has been defined on the identity column.

Procedure:

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 given values that 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.

Related concepts:

- "Comparing IDENTITY columns and sequences" on page 100

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

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.

Restrictions:

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.

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.

There are several restrictions on where NEXTVAL or PREVVAL expressions can be used.

Procedure:

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.

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 application process.

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, and all instances of NEXTVAL return the same value for a 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 row, and a PREVVAL expression for any additional rows.

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)
```

Related concepts:

- "Comparing IDENTITY columns and sequences" on page 100

Related reference:

- "CREATE SEQUENCE statement" in the *SQL Reference, Volume 2*

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.

Related tasks:

- "Defining a generated column on a new table" on page 95
- "Creating a sequence" on page 99
- "Defining a generated column on an existing table" on page 178

Examples of range-clustered tables

The two examples that follow are simple and demonstrate the ways to create a range-clustered table. The examples show how you can use either a single column, or multiple columns, as the key to the table. In addition, they show how to create a table that allows data to overflow and a table that does not allow data to overflow.

The first example shows a range-clustered table that is used to locate a student using a STUDENT_ID. For each student record, we include the following information:

- School ID
- Program ID
- Student number
- Student ID
- Student first name
- Student last name
- Student grade point average (GPA)

In this case, we would like to work with student records solely based on the STUDENT_ID. The STUDENT_ID will be used to add, update, and delete student records.

Note: Other indexes can be added separately at another time. However, for the purpose of this example, the organization of the table and how to access the table's data are defined when the table is created.

Here is the syntax needed for this table:

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    PROGRAM_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
```

```
   ORGANIZE BY KEY SEQUENCE
   (STUDENT_ID     STARTING FROM 1 ENDING AT 1000000)
    ALLOW OVERFLOW
;
```

The size of each record is the sum of the columns. In this case, there is a 10 byte header + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (for nullable columns) equaling 97 bytes. With a 4 KB page size (or 4096 bytes), after accounting for the overhead there is 4038 bytes, or enough room for 42 records per page. If we allow for 1 million student records, there will be a need for 1 million divided by 42 records per page, or 23809.5 pages. This rounds up to 23810 pages that are needed. Four pages are added for table overhead and three pages for extent mapping. The result is a required preallocation of 23817 pages of 4 KB size. (The extent mapping assumes a single container to hold this table. There should be three pages for each container.)

In the second example, which is a variation on the first, consider the idea of a school board. In the school board there are 200 schools, each having 20 classrooms with a capacity of 35 students. This school board can accommodate a maximum of 140,000 students.

In this case, we would like to work with student records based on three factors: the SCHOOL_ID, the CLASS_ID, and the STUDENT_NUM values. Each of these three columns will have unique values and will be used together to add, update, and delete student records.

Note: As with the previous example, other indexes may be added separately and at some other time.

Here is the syntax needed for this table:

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    CLASS_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (SCHOOL_ID      STARTING FROM 1 ENDING AT 200,
    CLASS_ID       STARTING FROM 1 ENDING AT 20,
    STUDENT_NUM    STARTING FROM 1 ENDING AT 35)
    DISALLOW OVERFLOW
;
```

In this case, an overflow is not allowed. This makes sense because there is likely a school board policy that restricts the number of students allowed in each class. In this example, the largest possible class size is 35. When you couple this factor with the physical limitations imposed by the number of classrooms and schools, it is clear that there is no reason to allow an overflow in the number of students in the school board.

It is possible that schools have varying numbers of classrooms. If this is the case, when defining the range for the number of classrooms (using CLASS_ID), the upper boundary should be the largest number of classrooms when considering all of the schools. This might mean that some smaller schools (schools with fewer classrooms than the largest school) will have space for student records that might never be used (unless, for example, portable classrooms are added to the school).

By using the same 4 KB page size and the same student record size as in the previous example, we know that there can be 42 records per page. With 140,000 student records, there will be a need for 3333.3 pages, or 3334 pages once rounding up is done. There are two pages for table information, and three pages for extent mapping. The result is a required preallocation of 3339 pages of 4 KB size.

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

How the SQL compiler works with range-clustered tables

The SQL compiler handles the range-clustered table (RCT) in a similar way to a regular table that has a secondary B+ tree index. Rather than working through a B+ tree index to determine the record's location or Record Identifier (RID), RCT uses a functional lookup involving the record key values and the algorithm from the range definition. This is similar to having an index because a key value is used to obtain the RID quickly.

When working to determine the best access path to required data, the SQL compiler uses statistical information kept about the tables. Index statistics are collected during a table scan when a RUNSTATS command is issued. For an RCT, the table is modeled as a regular table, and the index is modeled as a function-based index.

Order of records in the table is not guaranteed when creating a range-clustered table allowing overflow.

Related concepts:

- "Range-clustered tables" in the *Administration Guide: Planning*
- "Guidelines for using range-clustered tables" on page 103

Guidelines for using range-clustered tables

When working with DB2® Universal Database and range-clustered tables (RCT), note the following guidelines:

- When defining the range of key values, the minimum value is optional; if it is not specified, then the default is one (1). Negative values are allowed for minimum and maximum values. When working with negative values, the minimum value must be stated explicitly. For example, ORGANIZE BY KEY SEQUENCE (F1 STARTING FROM -100 ENDING AT -10)
- Creating a regular index on the same key values used to define the range-clustered table is not allowed.
- Some ALTER TABLE options are unavailable for use on range-clustered tables. Where the option does not affect the physical structure of the table, the option is allowed.
- Because the process of creating a range-clustered table preallocates the required disk space, that space must be available or the table creation will fail.

Related concepts:

- "Range-clustered tables" in the *Administration Guide: Planning*
- "Examples of range-clustered tables" on page 101

Defining dimensions on a table

A *dimension* is a clustering key for a table. One or more dimensions can be selected for a table. When you have more than one dimension on a table, it is considered to be a multidimensionally clustered table. Such a table is created using the CREATE TABLE statement with the ORGANIZE BY DIMENSIONS clause.

Restrictions:

The set of columns used in the ORGANIZE BY [DIMENSIONS] clause must follow the rules for the CREATE INDEX statement. The columns are treated as keys used to maintain the physical order of data in storage.

Procedure:

Each dimension is specified in the CREATE TABLE statement using the ORGANIZE BY [DIMENSIONS] clause and one or more columns. Parentheses are used within the dimension list to group columns to be associated with a single dimension.

Data is physically clustered on one or more dimensions, for as many dimensions as are specified, simultaneously. Data is organized by extent or "block" along dimension lines. When querying data using dimension predicates, the scan can be limited to only those extents of the table containing the dimension values involved. Further, since extents are sets of sequential pages on disk, very efficient prefetching can be performed for these scans.

Although a table with a single clustering index can become unclustered over time as space in the table is filled in, a table with multiple dimensions is able to maintain its clustering over all dimensions automatically and continuously. As a result, there is no need to reorganize the table in order to restore sequential order to the data.

A dimension block index is automatically created for each dimension specified. The dimension block index is used to access data along a dimension. The dimension block index points to extents instead of individual rows, and so are much smaller than regular indexes. These dimension block indexes can be used to very quickly access only those extents of the table that contain particular dimension values.

A composite block index is automatically created containing all dimension key columns. The composite block index is used to maintain the clustering of data during insert and update activity. The composite block index is used in query processing to access data in the table having particular dimension values.

Note: The order of key parts in the composite block index may affect its use or applicability for query processing. The order of its key parts is determined by the order of columns found in the entire ORGANIZE BY [DIMENSIONS] clause used when creating the MDC table. For example, if a table is created using:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

then the composite block index will be created on columns (c1,c4,c3,c2). Although c1 is specified twice in the dimensions clause, it is used only once as a key part for the composite block index, and in the order in which it is first found. The order of key parts in the composite block index makes no

difference for insert processing, but may do so for query processing. If it is more desirable, therefore, to have the composite block index with a column order (c1,c2,c3,c4), then the table should be created using:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

A composite block index is not created in the case where a specified dimension already contains all the columns that the composite block index would have. For example, a composite block index would not be created for the following table:

```
CREATE TABLE t1 (c1 int, c2 int)
   ORGANIZE BY DIMENSIONS (c1,(c2,c1))
```

Related concepts:

- "Multidimensional clustering tables" in the *Administration Guide: Planning*

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Creating a hierarchy table or a typed 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.

As part of creating a structured type hierarchy, you will create typed tables. You can use typed tables to store instances of objects whose characteristics are defined with the CREATE TYPE statement.

Prerequisites:

The type on which the hierarchy table or typed table will be created must exist.

Procedure:

You can create a hierarchy table or typed table using a variant of the CREATE TABLE statement.

Related concepts:

- "Typed tables" in the *Application Development Guide: Programming Server Applications*

Related tasks:

- "Populating a typed table" on page 106
- "Creating a typed view" on page 121
- "Creating a structured type hierarchy" in the *Application Development Guide: Programming Server Applications*
- "Dropping typed tables" in the *Application Development Guide: Programming Server Applications*
- "Creating typed tables" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

- "CREATE TYPE (Structured) statement" in the *SQL Reference, Volume 2*

Populating a typed table

As part of establishing a structured type hierarchy, you will create typed tables. Typed tables are used to store instances of objects whose characteristics are defined with the CREATE TYPE statement. Once created, you will need to place data into the typed table.

Prerequisites:

The typed table must exist.

Procedure:

You can populate a typed table after creating the structured types and then creating the corresponding tables and subtables.

Related concepts:

- "Substitutability in typed tables" in the *Application Development Guide: Programming Server Applications*
- "Typed tables" in the *Application Development Guide: Programming Server Applications*

Related tasks:

- "Creating a hierarchy table or a typed table" on page 105
- "Storing objects in typed table rows" in the *Application Development Guide: Programming Server Applications*
- "Dropping typed tables" in the *Application Development Guide: Programming Server Applications*
- "Creating typed tables" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "CREATE TYPE (Structured) statement" in the *SQL Reference, Volume 2*

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.

Prerequisites:

All table spaces must exist before the CREATE TABLE statement is run.

Restrictions:

The separation of the parts of the table can only be done using DMS table spaces.

Procedure:

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** 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.

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Creating a table in a partitioned database

There are performance advantages to creating a table across several partitions in a partitioned database. The work associated with the retrieval of data can be divided amoung the database partitions.

Prerequisites:

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 database partition group.
- 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 database partition group.

Restrictions:

You must be careful and 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).

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.

Procedure:

Creating a table that will be a part of several database partitions is specified when you are creating the table. There is an additional option 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.

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 database partition groups).

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.

Related concepts:

- "Database partition groups" in the *Administration Guide: Planning*
- "Database partition group design" in the *Administration Guide: Planning*
- "Table collocation" in the *Administration Guide: Planning*

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

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 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.

Restrictions:

You cannot use triggers with nicknames.

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 Editor (which replaces 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.

Procedure:

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
   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.

Related concepts:

- "Trigger dependencies" on page 111
- "INSERT, UPDATE, and DELETE triggers" in the *Application Development Guide: Programming Server Applications*
- "Triggers in application development" in the *Application Development Guide: Programming Server Applications*
- "Trigger creation guidelines" in the *Application Development Guide: Programming Server Applications*
- "Using triggers to update view contents" on page 111

Related tasks:

- "Dropping a trigger" on page 190
- "Creating triggers" in the *Application Development Guide: Programming Server Applications*
- "Defining business rules using triggers" in the *Application Development Guide: Programming Server Applications*
- "Defining actions using triggers" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "CREATE TRIGGER statement" in the *SQL Reference, Volume 2*

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 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.

Related concepts:

- "Using triggers to update view contents" on page 111

Related tasks:

- "Creating a trigger" on page 109
- "Dropping a trigger" on page 190

Related reference:

- "CREATE TRIGGER statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*

Using triggers to update view contents

INSTEAD OF triggers can be used to perform a delete, insert, or update request on behalf of a view which is not inherently updateable. Applications taking advantage of this type of trigger are able to write update operations against views just as if the view were a table.

For example, you could use the following SQL statements to create a view:

```
CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE,
      DEPTNAME)
   AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE, DEPTNAME
      FROM EMPLOYEE, DEPARTMENT WHERE EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
```

Due to the join in EMPV view's body, we would not be able to use the view to update data in the underlying tables until the following statements are added:

```
CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
   REFERENCING NEW AS NEWEMP DEFAULTS NULL FOR EACH ROW
   INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
      PHONENO, HIREDATE)
   VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
       COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
       WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
       RAISE_ERROR('70001', 'Unknown department name')),
       PHONENO, HIREDATE)
```

This CREATE TRIGGER statement will allow INSERT requests against EMPV view to be carried out.

```
CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
   REFERENCING OLD AS OLDEMP FOR EACH ROW
   DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

This CREATE TRIGGER statement will allow DELETE requests against EMPV view to be carried out.

```
CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
   REFERENCING NEW AS NEWEMP
               OLD AS OLDEMP
   DEFAULTS NULL FOR EACH ROW
   BEGIN ATOMIC
      VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
                  ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
      UPDATE EMPLOYEE AS E
         SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE) =
             (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
              COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                           WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
              RAISE_ERROR('70001', 'Unknown department name')),
              NEWEMP.PHONENO, NEWEMP.HIREDATE)
         WHERE NEWEMP.EMPNO = E.EMPNO;
   END
```

This CREATE TRIGGER statement will allow UPDATE requests against EMPV view to be carried out.

Related tasks:

- "Creating a trigger" on page 109

Related reference:

- "CREATE TRIGGER statement" in the *SQL Reference, Volume 2*

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.

Related concepts:

- "Scalar functions" in the *SQL Reference, Volume 1*
- "User-defined functions" in the *SQL Reference, Volume 1*
- "Table functions" in the *SQL Reference, Volume 1*
- "Creating a function mapping" on page 113
- "Statistics for user-defined functions" in the *Administration Guide: Performance*
- "General rules for updating catalog statistics manually" in the *Administration Guide: Performance*
- "DB2 User-Defined Functions and Methods" in the *Application Development Guide: Programming Client Applications*

Related tasks:

- "Creating a function template" on page 114

Related reference:

- "Functions" in the *SQL Reference, Volume 1*
- "CREATE FUNCTION statement" in the *SQL Reference, Volume 2*

Details on creating a user-defined function (UDF) or method

This sections gives information on federated considerations when creating user-defined functions or methods.

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 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.

Related concepts:

- "Host language program mappings with transform functions" in the *Application Development Guide: Programming Server Applications*

Related tasks:

- "Creating a function template" on page 114

Related reference:

- "CREATE FUNCTION MAPPING statement" in the *SQL Reference, Volume 2*

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.

Restrictions:

There is no local executable for the function, therefore 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.

Procedure:

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')
```

Related concepts:

- "Creating a function mapping" on page 113

Related reference:

- "CREATE FUNCTION (Sourced or Template) statement" in the *SQL Reference, Volume 2*

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.

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.

Related concepts:

- "Creating a user-defined structured type" on page 117

Related tasks:

- "Creating a user-defined distinct type" on page 116

Related reference:

- "User-defined types" in the *SQL Reference, Volume 1*
- "Data types" in the *SQL Reference, Volume 1*

Details on creating a user-defined type (UDT)

Distinct types and structured type definitions are discussed here as are federated type mappings.

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.

Restrictions:

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.

Procedure:

The following SQL statement creates the distinct type `t_educ` as a smallint:

```
CREATE DISTINCT TYPE T_EDUC AS SMALLINT WITH COMPARISONS
```

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.

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.

Related concepts:

- "User-defined type (UDT)" on page 115

Related reference:

- "CREATE DISTINCT TYPE statement" in the *SQL Reference, Volume 2*
- "CREATE TRANSFORM statement" in the *SQL Reference, Volume 2*
- "CREATE METHOD statement" in the *SQL Reference, Volume 2*
- "CREATE FUNCTION (Sourced or Template) statement" in the *SQL Reference, Volume 2*
- "User-defined types" in the *SQL Reference, Volume 1*
- "Data types" in the *SQL Reference, Volume 1*

Creating a user-defined structured type

A structured type is a user-defined data type that contains one or more named attributes. Each attribute has a name and a data type of its own. Attributes are properties that describe an instance of a type. 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.

Related concepts:

- "User-defined structured types" in the *Application Development Guide: Programming Server Applications*
- "Structured type hierarchies" in the *Application Development Guide: Programming Server Applications*

Related tasks:

- "Creating structured types" in the *Application Development Guide: Programming Server Applications*
- "Creating a structured type hierarchy" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "CREATE TYPE (Structured) statement" in the *SQL Reference, Volume 2*
- "User-defined types" in the *SQL Reference, Volume 1*

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.

Restrictions:

You cannot create a type mapping for a LOB, LONG VARCHAR/VARGRAPHIC, DATALINK, structured or distinct type.

Procedure:

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)
```

Related reference:

- "CREATE TYPE MAPPING statement" in the *SQL Reference, Volume 2*
- "Data Type Mappings between DB2 and OLE DB" in the *Application Development Guide: Programming Client Applications*

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.

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.

Prerequisites:

The base table, nickname, or view on which the view is to be based must already exist before the view can be created.

Restrictions:

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.

Procedure:

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.

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
```

Related concepts:

- "Views" in the *SQL Reference, Volume 1*
- "Table and view privileges" on page 229
- "Controlling access to data with views" on page 238
- "Using triggers to update view contents" on page 111

Related tasks:

- "Creating a typed view" on page 121
- "Removing rows from a table or view" on page 170
- "Altering or dropping a view" on page 192
- "Recovering inoperative views" on page 194

Related reference:

- "CREATE VIEW statement" in the *SQL Reference, Volume 2*
- "INSERT statement" in the *SQL Reference, Volume 2*

Details on creating a view

A typed view is based on a predefined structured type.

Creating a typed view

Procedure:

You can create a typed view using the CREATE VIEW statement.

Related concepts:

- "Typed views" in the *Application Development Guide: Programming Server Applications*

Related tasks:

- "Creating typed views" in the *Application Development Guide: Programming Server Applications*
- "Altering typed views" in the *Application Development Guide: Programming Server Applications*
- "Dropping typed views" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "CREATE VIEW statement" in the *SQL Reference, Volume 2*

Creating a materialized query table

A *materialized query table* is a table whose definition is based on the result of a query. As such, the materialized query 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 materialized query table than the base table or tables, the query executes against the materialized query table, and you obtain the result faster than you otherwise would.

Restrictions:

Materialized query tables defined with REFRESH DEFERRED are not used to optimize static SQL.

Setting the CURRENT REFRESH AGE special register to a value other than zero should be done with caution. By allowing a materialized query 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.

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 materialized query 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 materialized query tables:

1. You cannot alter a materialized query table.
2. You cannot alter the length of a column for a base table if that table has a materialized query table.
3. You cannot import data into a materialized query table.
4. You cannot create a unique index on a materialized query table.
5. You cannot create a materialized query table based on the result of a query that references one or more nicknames.

Procedure:

The creation of a materialized query table with the replication option can be used to replicate tables across all nodes in a partitioned database environment. These are known as "replicated materialized query tables".

In general a materialized query table, or a replicated materialized query table, is used for optimization of a query if the isolation level of the materialized query table, or the replicated materialized query 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 materialized query tables, and replicated materialized query tables, that are defined under CS or higher isolation levels are used for optimization.

To create a materialized query table, you use the CREATE 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 materialized query 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 creating a materialized query table, you have the option of specifying whether the system will maintain the materialized query table or the user will maintain the materialized query table. The default is system-maintained, which can be explicitly specified using the MAINTAINED BY SYSTEM clause. User-maintained materialized query tables are specified using the MAINTAINED BY USER clause.

If you create a system-maintained materialized query table, you have a further option of specifying whether the materialized query table is refreshed automatically when the base table is changed, or whether it is refreshed by using the REFRESH TABLE statement. To have the materialized query table refreshed

automatically when changes are made to the base table or tables, specify the REFRESH IMMEDIATE keyword. An immediate refresh is useful when:

- Your queries need to ensure the data they access is the most current
- The base table or tables are infrequently changed
- The refresh is not expensive.

The materialized query table, in this situation, can provide pre-computed results. If you want the refresh of the materialized query table to be deferred, specify the REFRESH DEFERRED keyword. Materialized query tables specified with REFRESH DEFERRED will **not** reflect changes to the underlying base tables. You should use materialized query tables where this is not a requirement. For example, if you run DSS queries, you would use the materialized query table to contain legacy data.

A materialized query 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.

You use the CURRENT REFRESH AGE special register to specify the amount of time that the materialized query table defined with REFRESH DEFERRED 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.

The CURRENT REFRESH AGE special register can be set to ANY, or a value of 99999999999999, to allow deferred materialized queries to be used in a dynamic query. 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). A value of zero (0) indicates that only materialized query tables defined with REFRESH IMMEDIATE may be used to optimize the processing of a query. In such a case, materialized query tables defined with REFRESH DEFERRED are not used for optimization.

Materialized query 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.

Materialized query tables have queries routed to them when the table has been defined using the ENABLE QUERY OPTIMIZATION clause, and, if a deferred materialized query table, the CURRENT REFRESH AGE special register has been set to ANY. However, with user-maintained materialized query tables, the use of the CURRENT REFRESH AGE special register is not the best method to control the rerouting of queries. The CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register will indicate which kind of cached data will be available for routing.

With activity affecting the source data, a materialized query table over time will no longer contain accurate data. You will need to use the REFRESH TABLE statement.

Related concepts:

- "Isolation levels" in the *SQL Reference, Volume 1*

Related tasks:

- "Altering materialized query table properties" on page 184
- "Refreshing the data in a materialized query table" on page 185
- "Dropping a materialized query or staging table" on page 194

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- "REFRESH TABLE statement" in the *SQL Reference, Volume 2*
- "SET CURRENT REFRESH AGE statement" in the *SQL Reference, Volume 2*
- "CURRENT REFRESH AGE special register" in the *SQL Reference, Volume 1*
- "CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register" in the *SQL Reference, Volume 1*
- "SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION statement" in the *SQL Reference, Volume 2*

Creating a user-maintained materialized query table

User-maintained materialized query tables (MQTs) are useful for database systems in which tables of summary data already exist. Custom applications that maintain such summary tables are common. Identifying existing summary tables as user-maintained MQTs causes the query optimizer to use the existing summary table to compute result sets for queries against the base tables.

Note: The query optimizer does not use user-maintained MQTs when selecting an access plan for static SQL queries.

Restrictions:

If you create a user-maintained materialized query table, the restrictions associated with a system-maintained materialized query table still apply but with the following exceptions:

- INSERT, UPDATE, and DELETE operations are allowed on the materialized query table. However, no validity checking is done against the underlying base tables. You are responsible for the correctness of the data.
- LOAD, EXPORT, IMPORT, and data replication will work with this type of materialized query table except there is no validity checking.
- You are not allowed to use the REFRESH TABLE statement on this type of materialized query table.
- You are not allowed to use the SET INTEGRITY ... IMMEDIATE CHECKED statement on this type of materialized query table.
- User-maintained materialized query tables must be defined as REFRESH DEFERRED.

See the "Creating a materialized query table" topic for additional restrictions.

Procedure:

To create a materialized query table, you use the CREATE TABLE statement with the AS *fullselect* clause and the IMMEDIATE or REFRESH DEFERRED options.

When creating a materialized query table, you have the option of specifying whether the system will maintain the materialized query table or the user will maintain the materialized query table. The default is system-maintained, which can be explicitly specified using the MAINTAINED BY SYSTEM clause. User-maintained materialized query tables are specified using the MAINTAINED BY USER clause.

In large database environments, or data warehouse environments, there are often custom applications that maintain and load user-maintained materialized query tables.

Note: For the optimizer to consider a user-maintained MQT, the query optimization level must be set at Level 2, or at a level greater than or equal to 5.

Related tasks:

- "Populating a user-maintained materialized query table" on page 125

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Populating a user-maintained materialized query table

Once you have created the table to hold the summary information, you will want to populate the materialized query table (MQT) with the summary data you want the optimizer to use when determining result sets.

Prerequisites:

Ensure that the table to hold the summary information exists.

Procedure:

You can populate user-maintained MQTs using triggers, insert operations, or the LOAD, IMPORT, and DB2 DataPropagator utilities. When performing the initial population of a user-maintained MQT, you can avoid logging overhead by using the LOAD or IMPORT utilities.

The following steps represent a typical approach for populating a user-maintained MQT:

- Make the base tables read-only to avoid the creation of new records or the modification of existing records.
- Extract the required data from the base tables and write it to an external file.
- Import or load the data from the external file into the MQT. You can use the LOAD or IMPORT utilities on a table in the CHECK PENDING NO ACCESS state.

  Note: If you want to populate the MQT with SQL insert operations, you need to reset the PENDING NO ACCESS state. However, the optimizer must first be disabled by using the DISABLE QUERY OPTIMIZATION option in the SET MATERIALIZED QUERY clause of the ALTER TABLE statement to ensure that a dynamic SQL query does not accidentally optimize to this MQT while the data in it is still being established. Once the MQT has been populated, optimization needs to be enabled using the ENABLE

QUERY OPTIMIZATION option in the SET MATERIALIZED QUERY clause of the ALTER TABLE statement.

- To issue SQL queries against a new MQT, reset the CHECK PENDING NO ACCESS state. By doing this, you indicate that you have assumed responsibility for data integrity of the materialized view. The statement to do this is:

```
DB2 SET INTEGRITY FOR example ALL IMMEDIATE UNCHECKED
```

- Make the base tables read/write.

Note: For the optimizer to consider a user-maintained MQT, the query optimization level must be set at Level 2, or at a level greater than or equal to 5.

Related reference:

- "IMPORT Command" in the *Command Reference*
- "LOAD Command" in the *Command Reference*

Creating a staging table

A *staging table* allows incremental maintenance support for deferred materialized query table. The staging table collects changes that need to be applied to the materialized query table to synchronize it with the contents of underlying tables. The use of staging tables eliminates the high lock contention caused by immediate maintenance content when an immediate refresh of the materialized query table is requested. Also, the materialized query tables no longer need to be entirely regenerated whenever a REFRESH TABLE is performed.

Materialized query tables are a powerful way to improve response time for complex queries, especially queries that might require some of the following operations:

- Aggregated data over one or more dimensions
- Joins and aggregates data over a group of tables
- Data from a commonly accessed subset of data
- Repartitioned data from a table, or part of a table, in a partitioned database environment

Restrictions:

Here are some of the key restrictions regarding staging tables:

1. The query used to define the staging table must be incrementally maintainable; that is, it must adhere to the same rules as a materialized query table with an immediate refresh option.
2. Only a deferred refresh can have a supporting staging table. The query also defines the materialized query table associated with the staging table. The materialized query table must be defined with REFRESH DEFERRED.
3. When refreshing using the staging tables, only a refresh to the current point in time is supported.

Procedure:

An inconsistent, incomplete, or pending state staging table cannot be used to incrementally refresh the associated materialized query table unless some other operations occur. These operations will make the content of the staging table

consistent with its associated materialized query table and its underlying tables, and to bring the staging table out of pending. Following a refresh of a materialized query table, the content of its staging table is cleared and the staging table is set to a normal state. A staging table may also be pruned intentionally by using the SET INTEGRITY statement with the appropriate options. Pruning will change the staging table to an inconsistent state. For example, the following statement forces the pruning of a staging table called STAGTAB1:

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

When a staging table is created, it is put in a pending state and has an indicator that shows that the table is inconsistent or incomplete with regard to the content of underlying tables and the associated materialized query table. The staging table needs to be brought out of the pending and inconsistent state in order to start collecting the changes from its underlying tables. While in a pending state, any attempts to make modifications to any of the staging table's underlying tables will fail, as will any attempts to refresh the associated materialized query table.

There are several ways a staging table may be brought out of a pending state; for example:

- SET INTEGRITY FOR <staging table name> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <staging table name> IMMEDIATE CHECKED

Related tasks:

- "Creating a materialized query table" on page 121
- "Altering materialized query table properties" on page 184
- "Refreshing the data in a materialized query table" on page 185
- "Dropping a materialized query or staging table" on page 194

Related reference:

- "SET INTEGRITY statement" in the *SQL Reference, Volume 2*

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.

Prerequisites:

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.

Restrictions:

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.

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.

Procedure:

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 OS/390 or z/Series employs two distinct concepts of aliases: ALIAS and SYNONYM. These two concepts differ from DB2 Universal Database as follows:

- ALIASes in DB2 for OS/390 or z/Series:
  - Require their creator to have special authority or privilege
  - Cannot reference other aliases.
- SYNONYMs in DB2 for OS/390 or z/Series:
  - 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.

Related concepts:

- "Aliases" in the *SQL Reference, Volume 1*

Related reference:

- "CREATE ALIAS statement" in the *SQL Reference, Volume 2*

Index, index extension, or 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.
- 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.

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 online. Online index defragmentation 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 index defragmentation, space could only be reclaimed by a reorganization of the data with or without including the index. When deciding whether to create an index with the ability to defragment index pages online, you should consider this question: Is the added performance cost of checking for space to merge each time a key is physically removed from a leaf page 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 a reorganization to reclaim space?

Notes:

1. Pages freed after an online index defragmentation 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 index defragmentation may not free up any non-leaf pages of the index, whereas a full reorganization will make the index as small as possible by reducing the non-leaf and leaf pages as well as the number of levels of the index.

2. In indexes created prior to Version 8, a key is physically removed from a leaf page as part of the deletion or update of a table row. For type 2 indexes, keys are just marked as deleted when a row is deleted or updated. It is not physically removed from a page until clean up is done some time after the deletion or update has commited. Such a clean up may be done by a subsequent transaction which is changing the page where the key is marked deleted. Clean up can be explicitly triggered using the CLEANUP ONLY [ALL | PAGES] option of the REORG INDEXES utility.

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 database partition group. Note that unique indexes defined in a multiple partition environment must be a superset of the partitioning key.

Related concepts:

- "Indexes" in the *SQL Reference, Volume 1*
- "Using an index" on page 133
- "Options on the CREATE INDEX statement" on page 133
- "Creating a user-defined extended index type" on page 137
- "Index privileges" on page 232

Related tasks:

- "Enabling parallelism when creating indexes" on page 10
- "Creating an index" on page 131
- "Renaming an existing table or index" on page 186
- "Dropping an index, index extension, or an index specification" on page 196

Related reference:

- "CREATE INDEX statement" in the *SQL Reference, Volume 2*
- "CREATE INDEX EXTENSION statement" in the *SQL Reference, Volume 2*

Details on creating an index, index extension, or index specification

You can work with the indexes maintained by the database manager, or you can specify your own index.

Creating an index

An *index* is a set of one or more keys, each pointing to rows in a table. An index allows more efficient access to rows in a table by creating a direct path to the data through pointers.

Procedure:

Performance Tip: If you are going to carry out the following series of tasks:

1. Create Table
2. Load Table
3. Create Index (without the COLLECT STATISTICS option)
4. Perform RUNSTATS

Or, if you are going to carry out the following series of tasks:
1. Create Table
2. Load Table
3. Create Index (with the COLLECT STATISTICS option)

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.

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 creating the table, you can choose to create a multi-dimensional clustering (MDC) table. By creating this type of table, block indexes are created. Regular indexes point to individual rows; block indexes point to blocks or extents of data, and are much smaller than regular indexes. Block indexes are stored, along with regular indexes, in the same table space.

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.

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>)
```

Related concepts:

- "Optimizing load performance" in the *Data Movement Utilities Guide and Reference*
- "Using an index" on page 133
- "Options on the CREATE INDEX statement" on page 133
- "Index privileges" on page 232

Related tasks:

- "Renaming an existing table or index" on page 186
- "Dropping an index, index extension, or an index specification" on page 196

Related reference:

- "CREATE INDEX statement" in the *SQL Reference, Volume 2*

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
- Uses columns with high cardinality

Related concepts:

- "Index planning tips" in the *Administration Guide: Performance*
- "Index performance tips" in the *Administration Guide: Performance*
- "Data access through index scans" in the *Administration Guide: Performance*
- "Table and index management for standard tables" in the *Administration Guide: Performance*
- "Table and index management for MDC tables" in the *Administration Guide: Performance*

Options on 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.

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 the index key values of these new rows are always new high key values for example, 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.

The above discussion also applies to new ″overflow″ rows that result from UPDATEs that increase the size of a row.

A single index created using the ALLOW REVERSE SCANS parameter on the CREATE INDEX statement can be scanned in a forward or a backward direction. That is, such indexes support scans in the direction defined when the index was created and scans in the opposite or reverse direction. The statement could look something like:

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

In this case, the index (iname) is formed based on descending values (DESC) in the given column (cname). By allowing reverse scans, although the index on the column is defined for scans in descending order, a scan can be done in ascending order (reverse order). The actual use of the index in both directions is not controlled by you but by the optimizer when creating and considering access plans.

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 defragmentation is enabled for this index. Once enabled, the following considerations are used to determine if an online index defragmentation takes place: After a key is physically removed 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 defragmentation enabled:

```
CREATE INDEX LASTN ON EMPLOYEE (LASTNAME) MINPCTUSED 20
```

When a key is physically removed from an index page of 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 CREATE INDEX statement allows you to create the index while, at the same time, allowing read and write access to the underlying table and any previously existing indexes. To restrict access to the table while creating the index, use the LOCK TABLE statement to lock the table before creating the index. The new index is created by scanning the underlying table. Any changes made to the table while the index is being created are logged. Once the new index is created, the changes are applied to the index. To apply the logged changes more quickly during the index creation, a separate copy of the changes is maintained in memory buffer space, which is allocated on demand from the utility heap. This allows the index creation to process the changes by directly reading from memory first, and reading through the logs, if necessary, at a much later time. Once all the changes have been applied to the index, the table is quiesced while the new index is made visible.

When creating a unique index, ensure that there are no duplicate keys in the table and that the concurrent inserts during index creation are not going to introduce duplicate keys. Index creation uses a deferred unique scheme to detect duplicate keys, and therefore no duplicate keys will be detected until the very end of index creation, at which point the index creation will fail because of the duplicate keys.

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.

The LEVEL2 PCTFREE clause directs the system to preserve a specified percentage of free space on each page in the second level of an index. You specify a percentage of free space when the index is created to accommodate future insertions and updates. The second level is the level immediately above the leaf level. The default is to preserve a minimum of 10 and the PCTFREE value in all non-leaf pages. The LEVEL2 PCTFREE parameter allows the default to be overwritten; if you use the LEVEL2 PCTFREE integer option in the CREATE INDEX statement, the integer percent of free space is left on level 2 intermediate pages. A minimum of 10 and the integer percent of free space is left on level 3 and higher intermediate pages. By leaving more free space on the second level, the number of page splits that occur at the second level of the index is reduced.

The PAGE SPLIT SYMMETRIC, PAGE SPLIT HIGH, and PAGE SPLIT LOW clauses allow a choice in the page split behavior when inserting into an index.

The PAGE SPLIT SYMMETRIC clause is a default page split behavior that splits roughly in the middle of an index page. Using this default behavior is best when the insertion into an index is random or does not follow one of the patterns that are addressed by the PAGE SPLIT HIGH and PAGE SPLIT LOW clauses.

The PAGE SPLIT HIGH behavior is useful when there are ever increasing ranges in the index. Increasing ranges in the index may occur when:

- There is an index with multiple key parts and there are many values (multiple index pages worth) where all except the last key part have the same value
- All inserts into the table would consist of a new value which has the same value as existing keys for all but the last key part
- The last key part of the inserted value is larger than that of the existing keys

For example, if we have the following key values in the index;

```
(1,1),(1,2),(1,3), ... (1,n),
(2,1),(2,2),(2,3), ... (2,n),
...
(m,1),(m,2),(m,3), ...(m,n)
```

then the next key to be inserted would have the value (x,y) where 1 <= x <= m and y > n. If the insertions follow such a pattern, the PAGE SPLIT HIGH clause can be used so that page splits do not result in many pages that are fifty percent empty.

Similarly, PAGE SPLIT LOW can be used when there are ever-decreasing ranges in the index, to avoid leaving pages 50 percent empty.

Note: If you want to add a primary or unique key, and you want the underlying index to use SPLIT HIGH, SPLIT LOW, PCTFREE, LEVEL2 PCTFREE, MINPCTUSED, CLUSTER, or ALLOW REVERSE SCANS you must first create an index specifying the desired keys and parameters. Then use an ALTER TABLE statement to add the primary or unique key. The ALTER TABLE statement will pick up and reuse the index that you have already created.

You can collect index statistics as part of the creation of the index. At the time when you use the CREATE INDEX statement, the key value statistics and the

physical statistics are available for use. By collecting the index statistics as part of the CREATE INDEX statement, you will not need to run the RUNSTATS utility immediately following the completion of the CREATE INDEX statement.

For example, the following SQL statement will collect basic index statistics as part of the creation of an index:

```
CREATE INDEX IDX1 ON TABL1 (COL1) COLLECT STATISTICS
```

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.

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.

In multiple partition databases, unique indexes must be defined as supersets of the partitioning key.

Related concepts:

- "Index performance tips" in the *Administration Guide: Performance*
- "Index reorganization" in the *Administration Guide: Performance*
- "Table and index management for standard tables" in the *Administration Guide: Performance*
- "Online index defragmentation" in the *Administration Guide: Performance*
- "Table and index management for MDC tables" in the *Administration Guide: Performance*

Related tasks:

- "Changing table attributes" on page 182

Related reference:

- "max_querydegree - Maximum query degree of parallelism configuration parameter" in the *Administration Guide: Performance*
- "intra_parallel - Enable intra-partition parallelism configuration parameter" in the *Administration Guide: Performance*
- "dft_degree - Default degree configuration parameter" in the *Administration Guide: Performance*
- "CREATE INDEX statement" in the *SQL Reference, Volume 2*

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.

Related concepts:

- "Details on index maintenance" on page 138
- "Details on index searching" on page 139
- "Details on index exploitation" on page 139
- "A scenario for defining an index extension" on page 141

Details on creating a user-defined extended index type

This section discusses the various aspects required when creating your own extended index type.

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.

Related concepts:

- "Creating a user-defined extended index type" on page 137

Related reference:

- "CREATE INDEX EXTENSION statement" in the *SQL Reference, Volume 2*

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 with or without including the 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.

Related concepts:

- "Creating a user-defined extended index type" on page 137
- "Details on index maintenance" on page 138
- "Details on index exploitation" on page 139

Related reference:

- "CREATE INDEX EXTENSION statement" in the *SQL Reference, Volume 2*

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.

Related concepts:

- "Creating a user-defined extended index type" on page 137
- "Details on index maintenance" on page 138

- "Details on index searching" on page 139
- "A scenario for defining an index extension" on page 141

Related reference:

- "CREATE FUNCTION (External Scalar) statement" in the *SQL Reference, Volume 2*

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.

Related concepts:

- "Creating a user-defined extended index type" on page 137
- "Details on index maintenance" on page 138
- "Details on index searching" on page 139
- "Details on index exploitation" on page 139

Related reference:

- "CREATE INDEX EXTENSION statement" in the *SQL Reference, Volume 2*
- "CREATE FUNCTION (External Scalar) statement" in the *SQL Reference, Volume 2*

Invoking the Configuration Advisor through the command line processor

Prerequisites:

The database is already created.

Procedure:

After creating the database, you can use the AUTOCONFIGURE command to invoke the Configuration Advisor. You can use this method even if you select the AUTOCONFIGURE option when creating the database.

You can use available options for AUTOCONFIGURE to define values for several configuration parameters and to determine the scope of the application of those parameters. The scope can be NONE, meaning none of the values are applied; DB ONLY, meaning only database configuration and buffer pool values are applied; or, DB AND DBM, meaning all parameters and their values are applied.

Related concepts:

- "Configuration parameters" in the *Administration Guide: Performance*

Related reference:

- "AUTOCONFIGURE Command" in the *Command Reference*

Chapter 5. Altering a database

This chapter focuses on what you must consider before altering a database; and, how to alter or drop database objects.

Altering an instance

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.

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.

As part of the management of licenses for your DB2 Universal Database™ (DB2 UDB) 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.

You should pay particular attention to the following:

- "Changing instances (UNIX only)"
- "Changing node and database configuration files" on page 149

Changing instances (UNIX only)

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® Universal Database (DB2 UDB) is updated by installing a Program Temporary Fix (PTF) or a patch, all the existing DB2 UDB instances should be updated using the **db2iupdt** 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.

Related concepts:

- "Instance creation" on page 15

Related tasks:

- "Updating instance configuration on UNIX" on page 146
- "Removing instances" on page 148

Related reference:

- "db2iupdt - Update Instances Command" in the *Command Reference*

Details on changing instances

Before changing an instance, you should list all of the existing instances.

Listing instances

Procedure:

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 **Instances** folder.
2. Right-click the Instances folder, and select **Add** from the pop-up menu.
3. On the Add Instance window, click **Refresh**.
4. Click the drop-down arrow to see a list of database instances.
5. Click **Cancel** to exit the window.

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 supported Windows platforms) use:

```
set db2instance
```

Updating instance configuration on UNIX

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.

Procedure:

The **db2iupdt** command is found in `/usr/opt/db2_08_01/instance/` directory on AIX. The **db2iupdt** command is found in `/opt/IBM/db2/V8.1/instance/` directory on HP-UX, Solaris Operating Environment, or Linux.

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, SERVER_ENCRYPT, or CLIENT. 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.
- –e

  Allows you to update each instance that exists. Those that exist can be shown using **db2ilist**.
- –u Fenced ID

  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: Fenced ID 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 Server Edition default instance type)
  - Client (DB2 client default instance type)

Examples:

- If you installed DB2 Universal Database Workgroup Server Edition or DB2 Universal Database Enterprise Server 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
  ```

Related tasks:

- "Removing instances" on page 148

Related reference:

- "db2ilist - List Instances Command" in the *Command Reference*
- "db2iupdt - Update Instances Command" in the *Command Reference*

Updating instance configuration on Windows

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.

Procedure:

The **db2iupdt** command is found in `\sqllib\bin` directory.

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`: hostname

  Overrides the default TCP/IP host name if there are one or more TCP/IP host names for the current machine.
- `/p`: instance profile path

  Specifies the new instance profile path for the updated instance.
- `/r`: baseport,endport

  Specifies the range of TCP/IP ports used by the partitioned database instance when running with multiple partitions.
- `/u`: username,password

  Specifies the account name and password for the DB2 service.

Related tasks:

- "Listing instances" on page 23
- "Updating instance configuration on UNIX" on page 146
- "Removing instances" on page 148

Removing instances

Procedure:

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.

Note: On UNIX operating systems, when attempting to drop an instance using the db2idrop command, a message is generated saying that the `sqllib` subdirectory cannot be removed, and in the `adm` subdirectory several files with the .nfs extension are being generated. The `adm` subdirectory is an NFS-mounted system and the files are controlled on the server. You must delete the *.nfs files from the fileserver from where the directory is being mounted. Then you can remove the `sqllib` subdirectory.

Related reference:

- "db2stop - Stop DB2 Command" in the *Command Reference*
- "TERMINATE Command" in the *Command Reference*
- "STOP DATABASE MANAGER Command" in the *Command Reference*
- "db2idrop - Remove Instance Command" in the *Command Reference*
- "db2ilist - List Instances Command" in the *Command Reference*

Changing node and database configuration files

To update the database configuration file, use the Configuration Advisor in the Control Center or run *db2 autoconfigure* with the appropriate options. The Configuration Advisor helps you to tune performance and to 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 are disconnected
- For database manager parameters, the next time that you stop and start the instance

In most cases, the values recommended by the Configuration Advisor will provide better performance than the default values because they are based on information about your workload and your own particular server. However, the values are designed to improve the performance of, though not necessarily optimize, your database system. Think of the values as a starting point on which you can make further adjustments to obtain optimized performance.

Prerequisites:

If you plan to change any database partition groups (adding or deleting partitions, or moving existing partitions), the node configuration file must be updated.

If you plan to change the database, you should review the values for the configuration parameters. You can adjust some values periodically as part of the ongoing changes made to the database that are based on how it is used.

Procedure:

To update 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 that you want to change, and click **Configuration Advisor**.
3. Click each page, and change information as required.
4. Click the **Results** page to review any suggested changes to the configuration parameters.
5. When you are ready to apply or save the updates, click **Finish**.

To use the Configuration Advisor from the command line, use the AUTOCONFIGURE command.

To update individual parameters in 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.

Related concepts:

- "Benchmark testing" in the *Administration Guide: Performance*

Related tasks:

- "Changing the database configuration across multiple partitions" on page 151
- "Configuring DB2 with configuration parameters" in the *Administration Guide: Performance*

Related reference:

- "GET DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*
- "UPDATE DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*

Changing the database configuration across multiple partitions

Procedure:

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.

Related concepts:

- "Issuing commands in a partitioned database environment" on page 343

Related tasks:

- "Changing node and database configuration files" on page 149

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" on page 152
- "Altering a database partition group" on page 152
- "Altering a table space" on page 153
- "Dropping a schema" on page 161
- Chapter 6, "Altering tables and other related table objects," on page 165
- "Altering a user-defined structured type" on page 185
- "Deleting and updating rows of a typed table" on page 186
- "Renaming an existing table or index" on page 186
- "Dropping a table" on page 188
- "Dropping a user-defined temporary table" on page 190
- "Dropping a trigger" on page 190
- "Dropping a user-defined function (UDF), function mapping, or method" on page 191
- "Dropping a user-defined type (UDT) or type mapping" on page 192
- "Altering or dropping a view" on page 192
- "Recovering inoperative views" on page 194
- "Dropping a materialized query or staging table" on page 194
- "Recovering inoperative summary tables" on page 195

- "Dropping an index, index extension, or an index specification" on page 196
- "Statement dependencies when changing objects" on page 197

Dropping a database

Procedure:

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.

Related reference:

- "GET SNAPSHOT Command" in the *Command Reference*
- "DROP DATABASE Command" in the *Command Reference*
- "LIST ACTIVE DATABASES Command" in the *Command Reference*

Altering a database partition group

Procedure:

Once you add or drop partitions, you must redistribute the current data across the new set of partitions in the database partition group. To do this, use the REDISTRIBUTE DATABASE PARTITION GROUP command.

Related concepts:

- "Data redistribution" in the *Administration Guide: Performance*
- "Management of database server capacity" in the *Administration Guide: Performance*

Related tasks:

- "Redistributing data across partitions" in the *Administration Guide: Performance*

Related reference:

- "REDISTRIBUTE DATABASE PARTITION GROUP Command" in the *Command Reference*

Altering a table space

Procedure:

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, nor create another one; and there must always be at least one system temporary table space with a page size of 4 KB. 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.

Related tasks:

- "Adding a container to a DMS table space" on page 153
- "Modifying containers in a DMS table space" on page 154
- "Adding a container to an SMS table space on a partition" on page 158
- "Renaming a table space" on page 158
- "Dropping a user table space" on page 159
- "Dropping a system temporary table space" on page 160
- "Dropping a user temporary table space" on page 161

Related reference:

- "ALTER TABLESPACE statement" in the *SQL Reference, Volume 2*

Details on altering a table space

This section reviews those tasks associated with altering table spaces.

Adding a container to a DMS table space

Procedure:

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.

When new containers are added to a table space, or existing containers are extended, a rebalance of the table space may occur. The process of rebalancing involves moving table space extents from one location to another. During this process, the attempt is made to keep data striped within the table space. Rebalancing does not necessarily occur across all containers but depends on many factors such as on the existing container configuration, the size of the new containers, and how full is the table space.

When containers are added to an existing table space, they may be added such that they do not start in stripe 0. Where they start in the map is determined by the database manager and is based on the size of the containers being added. If the

container being added is not large enough, it is positioned such that it ends in the last stripe of the map. If it is large enough, it is positioned to start in stripe 0.

No rebalancing occurs if you are adding new containers and creating a new stripe set. A new stripe set is created using the BEGIN NEW STRIPE SET clause on the ALTER TABLESPACE statement. You can also add containers to existing stripe sets using the ADD TO STRIPE SET clause on the ALTER TABLESPACE statement.

Access to the table space is not restricted during the rebalancing. 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**.

To add a container to a DMS table space using the command line, enter:

```
ALTER TABLESPACE <name>
   ADD (DEVICE '<path>' <size>, FILE '<filename>' <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.

Related concepts:

- "Table space impact on query optimization" in the *Administration Guide: Performance*
- "How containers are added and extended in DMS table spaces" in the *Administration Guide: Planning*

Related tasks:

- "Adding a container to an SMS table space on a partition" on page 158

Related reference:

- "ALTER TABLESPACE statement" in the *SQL Reference, Volume 2*

Modifying containers in a DMS table space

Restrictions:

Each raw device can only be used as one container. The raw device size is fixed after its creation. When you are considering to use the resize or extend options to increase a raw device container, you should check the raw device size first to ensure that you do not attempt to increase the device container size larger than the raw device size.

Procedure:

You can resize the containers in a DMS table space (that is, one created with the MANAGED BY DATABASE clause).

To increase the size of one or more containers in 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 **Resize**, complete the information, and click **Ok**.

You can also drop existing containers from a DMS table space, reduce the size of existing containers in a DMS table space, and add new containers to a DMS table space without requiring a rebalance of the data across all of the containers.

The dropping of existing table space containers as well as the reduction in size of existing containers is only allowed if the number of extents being dropped or reduced in size is less than or equal to the number of free extents above the high-water mark in the table space. The high-water mark is the page number of the highest allocated page in the table space. This mark is not the same as the number of used pages in the table space because some of the extents below the high-water mark may have been made available for reuse.

The number of free extents above the high-water mark in the table space is important because all extents up to and including the high-water mark must sit in the same logical position within the table space. The resulting table space must have enough space to hold all of the data. If there is not enough free space, an error message (SQL20170N, SQLSTATE 57059) will result.

To drop containers, the DROP option is used on the ALTER TABLESPACE statement. For example:

```
ALTER TABLESPACE TS1 DROP (FILE 'file1', DEVICE '/dev/rdisk1')
```

To reduce the size of existing containers, you can use either the RESIZE option or the REDUCE option. When using the RESIZE option, all of the containers listed as part of the statement must either be increased in size, or decreased in size. You cannot increase some containers and decrease other containers in the same statement. You should consider the resizing method if you know the new lower limit for the size of the container. You should consider the reduction method if you do not know (or care about) the current size of the container.

To decrease the size of one or more containers in a DMS table space using the command line, enter:

```
ALTER TABLESPACE <name>
   REDUCE (FILE '<filename>' <size>)
```

The following example illustrates how to reduce a file container (which already exists with 1 000 pages) in a table space on a Windows-based system:

```
ALTER TABLESPACE PAYROLL
  REDUCE (FILE 'd:\hldr\finance'  200)
```

Following this action, the file is decreased from 1 000 pages in size to 800 pages.

To increase the size of 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. The contents of the table space may be rebalanced across the containers. Access to the table space is not restricted during the rebalancing.

To extend one or more containers in a DMS table space using the command line, enter:

```
ALTER TABLESPACE <name>
   EXTEND (FILE '<filename>' <size>)
```

The following example illustrates how to increase file containers (each already existing with 1 000 pages) in a table space on a Windows-based system:

```
ALTER TABLESPACE PERSNEL
  EXTEND (FILE 'e:\wrkhist1'  200
          FILE  'f:\wrkhist2'  200)
```

Following this action, the two files have increased from 1 000 pages in size to 1 200 pages. The contents of the table space may be rebalanced across the 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. The number of prefetchers is controlled by the *num_ioservers* database configuration parameter. The database has to be stopped for the new parameter value to take effect. In other words, all applications and users must disconnect from the database for the change to take affect.

Note that the ALTER TABLESPACE statement allows you to change other properties of the table space that can affect performance.

Related reference:

- "ALTER TABLESPACE statement" in the *SQL Reference, Volume 2*

Automatic prefetchsize adjustment after adding or dropping containers

If there is the possibility that you might forget to update the prefetch size of a table space after either adding or dropping containers, you should consider allowing the prefetch size to be determined by the database manager automatically.

If you forget to update the prefetch size, then there may be a noticeable degradation in the performance of the database.

DB2® Universal Database (DB2 UDB) is set up so that the automatic prefetch size is the default for any table spaces created using Version 8.2 (and later). The database manager uses the following formula to calculate the prefetch size for the table space:

```
prefetch size = (number of containers) X (number of physical spindles per
                    container) X extent size
```

There are three ways not to have the prefetch size of the table space set at AUTOMATIC:

- Create the table space with a specific prefetch size. Manually choosing a value for the prefetch size indicates that you will remember to adjust, if necessary, the prefetch size whenever there is an adjustment in the number of containers associated with the table space.
- Do not use prefetch size when creating the table space, and have the *dft_prefetch_sz* database configuration parameter set to a non-AUTOMATIC value. DB2 UDB checks this parameter when there is no explicit mention of the prefetch size when creating the table space. If a value other than AUTOMATIC is found, then that value is what is used as the default prefetch size. And you will need to remember to adjust, if necessary, the prefetch size whenever there is an adjustment in the number of containers associated with the table space.
- Alter the prefetch size manually by using the ALTER TABLESPACE statement.

Use of DB2_PARALLEL_IO

Prefetch requests are broken down into several smaller prefetch requests based on the parallelism of a table space, and before the requests are submitted to the prefetch queues. The DB2_PARALLEL_IO registry variable is used to define the number of physical spindles per container as well as influencing the parallel I/O on the table space. With parallel I/O off, the parallelism of a table space is equal to the number of containers. With parallel I/O on, the parallelism of a table space is equal to the number of container multiplied by the value given in the DB2_PARALLEL_IO registry variable. (Another way of saying this is, the parallelism of the table space is equal to the prefetch size divided by the extent size of the table space.)

Here are several examples of how the DB2_PARALLEL_IO registry variable influences the prefetch size. (Assume all of the following table spaces have been defined with an AUTOMATIC prefetch size.)

- DB2_PARALLEL_IO=*
  - All table spaces will use the default where the number of spindles equals 6 for each container. The prefetch size will be 6 times larger with parallel I/O on.
  - All table spaces will have parallel I/O on. The prefetch request is broken down to several smaller requests, each equal to the prefetch size divided by the extent size (or equal to the number of containers times the number of spindles).
- DB2_PARALLEL_IO=*:3
  - All table spaces will use 3 as the number of spindles per container.
  - All table spaces will have parallel I/O on.
- DB2_PARALLEL_IO=*:3,1:1

- All table spaces will use 3 as the number of spindles per container except for table space 1 which will use 1.
- All table spaces will have parallel I/O on.

Related tasks:

- "Altering a table space" on page 153
- "Adding a container to a DMS table space" on page 153
- "Modifying containers in a DMS table space" on page 154

Adding a container to an SMS table space on a partition

Restrictions:

You can only add a container to a SMS table space on a partition that currently has no containers.

Procedure:

To add a container to an SMS table space using the command line, enter the following:

```
ALTER TABLESPACE <name>
   ADD ('<path>')
   ON DBPARTITIONNUM (<partition_number>)
```

The partition specified by number, and every partition (or node) in the range of partitions, must exist in the database partition group on which the table space is defined. A partition_number may only appear explicitly or within a range in exactly one *db-partitions-clause* for the statement.

The following example shows how to add a new container to partition number 3 of the database partition group used by table space "plans" on a UNIX based operating system:

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON DBPARTITIONNUM (3)
```

Related tasks:

- "Adding a container to a DMS table space" on page 153
- "Modifying containers in a DMS table space" on page 154

Related reference:

- "ALTER TABLESPACE statement" in the *SQL Reference, Volume 2*

Renaming a table space

Restrictions:

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.

Procedure:

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.

Related reference:

- "RENAME TABLESPACE statement" in the *SQL Reference, Volume 2*

Switching the state of a table space

Procedure:

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:

```
db2 ALTER TABLESPACE <name>
   SWITCH ONLINE
```

Related reference:

- "ALTER TABLESPACE statement" in the *SQL Reference, Volume 2*

Dropping a user table space

Procedure:

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.

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
```

Related tasks:

- "Dropping a system temporary table space" on page 160
- "Dropping a user temporary table space" on page 161

Related reference:

- "COMMIT statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*

Dropping a system temporary table space

Restrictions:

You cannot drop a system temporary table space that has a page size of 4 KB without first creating another system temporary table space. The new system temporary table space must have a page size of 4 KB because the database must always have at least one system temporary table space that has a page size of 4 KB. For example, if you have a single system temporary table space with a page size of 4 KB, and you wish to add a container to it, and it is an SMS table space, you must first add a new 4 KB pagesize system temporary table space with the proper number of containers, and then drop the old system temporary table space. (If you were using DMS, you could add a container without having to drop and recreate the table space.)

Procedure:

The default table space page size is 4 KB.

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**.

This is the statement to create a system temporary table space:

```
CREATE SYSTEM TEMPORARY TABLESPACE <name>
   MANAGED BY SYSTEM USING ('<directories>')
```

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:\systemp2')
```

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.

Related tasks:

- "Dropping a user table space" on page 159
- "Dropping a user temporary table space" on page 161

Related reference:

- "CREATE TABLESPACE statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*

Dropping a user temporary table space

Procedure:

You can only drop a user temporary table space if there are no declared temporary tables currently 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.

Related tasks:

- "Dropping a user table space" on page 159
- "Dropping a system temporary table space" on page 160

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Dropping a schema

Procedure:

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.

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Altering a buffer pool

You might need to complete one of the following tasks when working with an existing buffer pool:

- Modify the size of the buffer pool on all partitons or on a single partition.
- Enable or disable the use of extended storage.
- Add this buffer pool definition to a new database partition group.
- Modify the block area of the buffer pool for block-based I/O.

Prerequisites:

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

Procedure:

1. `SELECT BPNAME FROM SYSCAT.BUFFERPOOLS` to get the list of the buffer pool names that already exist in the database.
2. Choose the buffer pool name from the result list.
3. Determine what changes need to be made.
4. Ensure that you have the correct authorization ID to run the ALTER BUFFERPOOL statement.

   Note: Two key parameters are `IMMEDIATE` and `DEFERRED`. With IMMEDIATE, the buffer pool size is changed without delay. If there is insufficient reserved space in the database shared memory to allocate new space, the statement is run as deferred.

   With DEFERRED, the buffer pool is cached when the database is reactivated following the disconnection of those applications. Reserved memory space is not needed; DB2 UDB allocates the required memory from the system at activation time.

5. Use the ALTER BUFFERPOOL statement to alter a single quality of the buffer pool object.

Related tasks:

- "Creating a buffer pool" on page 64

Related reference:

- "ALTER BUFFERPOOL statement" in the *SQL Reference, Volume 2*

Chapter 6. Altering tables and other related table objects

Tasks that are required for modifying the structure and content of the table and related table objects include the following:

- "Space compression for existing tables"
- "Adding columns to an existing table" on page 168
- "Modifying a column definition" on page 169
- "Removing rows from a table or view" on page 170
- "Updating table and view contents using the MERGE statement" on page 187
- "Modifying an identity column definition" on page 171
- "Altering a constraint" on page 172
- "Defining a generated column on an existing table" on page 178
- "Declaring a table volatile" on page 180
- "Changing partitioning keys" on page 181
- "Changing table attributes" on page 182
- "Altering materialized query table properties" on page 184
- "Refreshing the data in a materialized query table" on page 185

Note that you cannot alter triggers for tables; you must drop any trigger that is no longer appropriate (see "Dropping a trigger" on page 190), and add its replacement (see "Creating a trigger" on page 109).

Modifying existing tables and their related table objects

Space compression for existing tables

An existing table can be changed to the record format that allows space compression. The sum of the byte counts of the columns in the record format allowing space compression may exceed the sum of the byte counts of the columns in the original record format (that does not allow space compression) as long as the sum of the byte counts does not exceed allowable row length of the table in the tablespace. For example, the allowable row length is 4005 bytes in a tablespace with 4 KB page size. If the allowable row length is exceeded, the error message SQL0670N is returned. The byte count formula is documented as part of the CREATE TABLE statement.

Similarly, an existing table can be changed from a record format that allows space compression to a record format that does not. The same condition regarding the sum of the byte counts of the columns applies; and the error message SQL0670N is returned as necessary.

To determine if you should consider space compression for your table, you should know that a table with the majority of values equal to the system default values, or NULL, would benefit from the new row format. For example, where there is an INTEGER column and 90% of the column has values of 0 (the default value for the data type INTEGER), or NULL, compressing this table plus this column would benefit from the new row format and save a lot of disk space.

When altering a table, you can use the VALUE COMPRESSION clause to specify that the table is using the space row format at the table level and possibly at the column level. You would use ACTIVATE VALUE COMPRESSION to specify that the table will use the space saving techniques or you would use DEACTIVATE VALUE COMPRESSION to specify that the table will no longer use space saving techniques for data in the table.

If you use DEACTIVATE VALUE COMPRESSION, this will implicitly disable any COMPRESS SYSTEM DEFAULT options associated with columns in that table.

After modifying the table to a new row format, all subsequent rows inserted, loaded, or updated will have the new row format. To have every row modified to the new row format, you should run a reorganization of the table or perform an update operation on existing rows before changing the row format.

Related concepts:

- "Space compression for new tables" on page 87

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Altering a table using a stored procedure

Tables are where all of the data for your business is stored. Before you create your database, you have to consider the type and organization of the data that you want to keep in the database. A lot of planning is required to ensure that you have thought to use and manipulate all of the relevant data that you and your business might need. However, things change. In spite of good planning, there may be new requirements or business changes that necessitate changes being made to the tables in the database.

You might find that you need to change in one or more of the following ways within your table:

- Rename columns
- Remove columns
- Alter column type and transform existing data using SQL scalar functions
- Increase or decrease column size
- Change column default value
- Change column from NOT NULL to NULLABLE
- Change precision and scale for decimal

When making these types of changes, you need to minimize the risk of losing the original table data. DB2® Universal Database (DB2 UDB) provides a user interface, and a stored procedure, that will allow you to alter a table. The original table and its associated data are not dropped until you explicitly indicate that all of the alter table work that you desire has been completed.

Each stored procedure call that is invoked from the user interface carries out a sequence of actions such as dropping, recreating, and loading data to accomplish the actions listed above.

There are limitations on what can be altered in the table. These limitations include:

- No support for altering materialized query tables (MQTs).

  However, there is support for altering a table which has a MQT. Also, MQTs defined on a base table which is altered is not refreshed (populated) during the ALTER TABLE process. In an MQT, while its base table is being altered by the ALTOBJ() stored procedure, all of the columns which are not part of the select result from the base table are lost because the MQT content is completely rebuilt from the new base table.
- No support for altering type tables, or a table that is the scope of any existing reference column-type table.
- No support for altering a remote table using a nickname.
- Column sequence within the table cannot be reordered.
- Add and rename are exclusive to drop column actions.

  That is, these column actions cannot coexist in one single alter table call.
- The DATALINK data type is not supported.
- The definition of the objects may change between ALTOBJ() calls because there are no object locks that persist.
- Table profiles, such as a runstats profile, that are associated with the table pack descriptor are lost after going through the ALTER TABLE process.
- Only one sequence of ALTER TABLE stored procedure calls is supported per table at any given time. That is, once the ALTOBJ() stored procedure is called, it should be finished or rolled back before another ALTER TABLE can be started on the same table. Altering multiple tables at the same time using the ALTOBJ() stored procedure is supported as long as the table dependencies do not collide.

There are several component pieces that make up the available options when using the stored procedure that carryout the ALTER TABLE actions. These pieces include:

- ALTER_OBJ('GENERATE','<sql statement>, 0, ?)

  This procedure generates all of the SQL statements and places them into a metadata table.

  Note: In generate mode, the SQL statement parameter cannot be null; and, if an alter ID is provided, it is ignored.
- ALTER_OBJ('VALIDATE',NULL,123,?)

  This procedure verifies the SQL generated but does not include the movement of data. The running of the scripts to test validity takes place under the given user ID "123". The results of the verification are placed in the Meta table (which also holds the other information from the table being altered).
- ALTER_OBJ('APPLY_CONTINUE_ON_ERROR',NULL,123,?)

  This procedure runs all of the SQL statements under the given ID, and writes the results into the Meta table. The SQL statements would include how to build the new table, the building of any dependent objects, and the populating of the new table.

  You can get the old definitions back using the UNDO mode (see below).

  A warning SQLCODE is set for the stored procedure in the SQLCA; and the transactions in the stored procedure are finished.
- ALTER_OBJ('APPLY_STOP_ON_ERROR',NULL,123,?)

  This procedure runs each of the SQL statements one-by-one under the given ID, and stops when any errors are encountered.

  An error SQLCODE is set for the stored procedure in the SQLCA; and the transactions in the stored procedure are automatically rolled back.

- ALTER_OBJ('UNDO',NULL,123,?)

  Run the script that contains all of the changes made by the alter table actions under the given user ID. All of those changes are undone.

  Note: When working with the ALTOBJ_UNDO, the ID parameter cannot be null.
- ALTER_OBJ('FINISH',NULL,123,?)

  This procedure deletes the original table, and cleans up all of the entries found in the Meta table under the given user ID.

  Note: This mode can only be called separately from all other modes.

Related reference:

- "Supported functions and SQL administrative routines" in the *SQL Reference, Volume 1*
- "ALTOBJ procedure" in the *SQL Administrative Routines*

Adding columns to an existing table

Procedure:

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)
```

Related tasks:

- "Modifying a column definition" on page 169

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Modifying a column definition

Procedure:

You can modify the characteristics of a column by increasing the length of an existing VARCHAR or VARGRAPHIC column. The number of characters may increase up to a value dependent on the page size used.

You can modify the default value associated with a column. Once you have defined the new default value, the new value is used for the column in any subsequent SQL operations where the use of the default is indicated. The new value must follow the rules for assignment and have the same restrictions as documented under the CREATE TABLE statement.

Note: Generate columns cannot have their default value altered by this statement.

To modify the length of a column 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 the length and type of a column of an existing table using the command line, enter:

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   <modification_type>
```

For example, to increase a column up to 4000 characters, use something similar to the following:

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DATA TYPE VARCHAR(4000)
```

In another example, to allow a column to have a new VARGRAPHIC value, use an SQL statement similar to the following:

```
ALTER TABLE t1
   ALTER COLUMN colnam2
   SET DATA TYPE VARGRAPHIC(2000)
```

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 t1
   ALTER COLUMN colnamt1
   ADD SCOPE typtab1
```

To modify the default value of a column of an existing table using the command line, enter:

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   SET DEFAULT 'new_default_value'
```

For example, to change the default value for a column, use something similar to the following:

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DEFAULT '123'
```

Related tasks:

- "Modifying an identity column definition" on page 171

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Removing rows from a table or view

Procedure:

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 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.

Related concepts:

- "Locks and concurrency control" in the *Administration Guide: Performance*
- "Locks and performance" in the *Administration Guide: Performance*
- "Factors that affect locking" in the *Administration Guide: Performance*
- "Guidelines for locking" in the *Administration Guide: Performance*

Related reference:

- "DELETE statement" in the *SQL Reference, Volume 2*

Modifying the generated or identity property of a column

You can add and drop the generated or identity property of a column in a table using the ALTER COLUMN clause in the ALTER TABLE statement.

You can do one of the following actions:

- When working with an existing non-generated column, you can add a generated expression attribute. The modified column then becomes a generated column.
- When working with an existing generated column, you can drop a generated expression attribute. The modified column then becomes a normal, non-generated column.
- When working with an existing non-identity column, you can add a identity attribute. The modified column then becomes an identity column.
- When working with an existing identity column, you can drop the identity attribute. The modified column then becomes a normal, non-generated, non-identity column.
- When working with an existing generated column, you can alter a generated column from being GENERATED ALWAYS to GENERATED BY DEFAULT. The reverse is also true; that is, you can alter a generated column from being GENERATED BY DEFAULT to GENERATED ALWAYS. This is only possible when working with a generated column.
- You can drop the default attribute from the user-defined default column. When you do this, the new default value is null.
- You can drop the default, identity, or generation attribute and then set a new default, identity, or generation attribute in the same ALTER COLUMN statement.
- For both the CREATE TABLE and ALTER TABLE statements, the "ALWAYS" is an optional word in the GENERATED clause. This means that GENERATED ALWAYS is equivalent to GENERATED when used in the ALTER TABLE statement.

Related tasks:

- "Defining a generated column on a new table" on page 95
- "Defining an identity column on a new table" on page 98

Modifying an identity column definition

Procedure:

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.

Related reference:

- "MAX aggregate function" in the *SQL Reference, Volume 1*
- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "LOAD Command" in the *Command Reference*

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 unique constraint" on page 175

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.

Adding a unique constraint

Procedure:

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)
```

Related tasks:

- "Defining a unique constraint" on page 89

- "Dropping a unique constraint" on page 175

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Adding primary keys

Procedure:

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>
```

Related tasks:

- "Adding foreign keys" on page 173

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "SET INTEGRITY statement" in the *SQL Reference, Volume 2*

Adding foreign keys

Procedure:

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.

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
```

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related tasks:

- "Adding primary keys" on page 173

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Adding a table check constraint

Procedure:

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.

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)
```

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "SET INTEGRITY statement" in the *SQL Reference, Volume 2*

Dropping a unique constraint

You drop constraints with the ALTER TABLE statement. For more information on this statement, including its syntax, refer to the *SQL Reference* manual.

Dropping a unique constraint

Procedure:

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.

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Dropping primary keys

Procedure:

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.

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related tasks:

- "Dropping a foreign key" on page 176

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Dropping a foreign key

Procedure:

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
```

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related tasks:

- "Dropping primary keys" on page 176

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Dropping a table check constraint

Procedure:

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. 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
```

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related tasks:

- "Adding a table check constraint" on page 174

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

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.

Prerequisites:

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.

Restrictions:

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.

Procedure:

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. Assign the correct values to the generated columns. This can be accomplished using the following methods:
   - Recompute and reassign the values for the generated columns using:

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATED
     ```

     If this SET INTEGRITY statement fails because of a lack of log space, then increase the available active log space and reissue the SET INTEGRITY statement.

     Note: Exception tables can be used at this point.
   - If it is not possible to increase the available active log space, then use searched update statements to assign the generated columns to their default values.
     a. Get an exclusive lock on the table. This prevents all but uncommitted read transactions from accessing the table. Note that the table lock will be released upon the first intermittent commit and other transactions will be able to see rows with generated columns that has not yet been assigned to their default values.

        ```
        LOCK TABLE t1
        ```

     b. Bypass checking of the generated columns

        ```
        SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
        ```

     c. Check the table for other integrity violations (if applicable) and bring it out of check pending

        ```
        SET INTEGRITY FOR t1 IMMEDIATE CHECKED
        ```

     d. Update the generated columns using intermittent commits and predicates to avoid the logs filling up.

        ```
        UPDATE t1 SET (c3, c4) = (DEFAULT, DEFAULT) WHERE <predicate>
        ```

     e. Unlock the table by completing the transaction using a commit statement.

        ```
        COMMIT
        ```
   - A cursor based approach may also be used if it is not possible to increase the available active log space:
     a. Declare a FOR UPDATE cursor for table. The WITH HOLD option should be used if locks should be retained after the intermittent commits.

        ```
        DECLARE C1 CURSOR WITH HOLD FOR S1
        ```

        Where S1 is defined as:

        ```
        SELECT '0' FROM t1 FOR UPDATE OF C3, C4
        ```

     b. Open the cursor.

        ```
        OPEN C1
        ```

     c. Bypass checking of the generated columns

        ```
        SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
        ```

     d. Check the table for other integrity violations (if applicable) and bring it out of check pending

        ```
        SET INTEGRITY FOR t1 IMMEDIATE CHECKED
        ```

     e. Have a loop to fetch all rows in the table and for each row fetched, execute the following to assign the generated columns to their default

tables. It is important to make sure that the first fetch is done right after the table is brought out of check pending to ensure that the table is locked for the duration of the cursor.

```
UPDATE t1 SET (C3, C4) = (DEFAULT, DEFAULT) WHERE CURRENT OF C1
```

Do intermittent commits to avoid the logs filling up.

f. Close the cursor and commit to unlock the table.

```
CLOSE C1
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 GENERATION
```

  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
```

Related tasks:

- "Defining a generated column on a new table" on page 95

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "COMMIT statement" in the *SQL Reference, Volume 2*
- "LOCK TABLE statement" in the *SQL Reference, Volume 2*
- "SET INTEGRITY statement" in the *SQL Reference, Volume 2*
- "UPDATE statement" in the *SQL Reference, Volume 2*
- "db2look - DB2 Statistics and DDL Extraction Tool Command" in the *Command Reference*

Declaring a table volatile

Procedure:

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
```

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Changing partitioning keys

Procedure:

You can only change a partitioning key on tables in single-partition database partition groups. 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
```

You cannot change the partitioning key of a table in a multiple partition database partition groups. If you try to drop it, an error is returned.

To change the partitioning key of multiple partition database partition groupss, either:

- Export all of the data to a single-partition database partition groups 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.

Related concepts:

- "Partitioning keys" in the *Administration Guide: Planning*

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Changing table attributes

Procedure:

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.

Related concepts:

- "Locks and concurrency control" in the *Administration Guide: Performance*
- "Lock attributes" in the *Administration Guide: Performance*
- "Locks and performance" in the *Administration Guide: Performance*
- "Factors that affect locking" in the *Administration Guide: Performance*
- "Guidelines for locking" in the *Administration Guide: Performance*

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Altering an identity column

Procedure:

Modify the attributes of an existing identity column with the ALTER TABLE statement.

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.

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*

Altering a sequence

Procedure:

Modify the attributes of an existing sequence with the ALTER SEQUENCE statement.

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.

Related tasks:

- "Dropping a sequence" on page 183

Related reference:

- "ALTER SEQUENCE statement" in the *SQL Reference, Volume 2*

Dropping a sequence

Procedure:

To delete a sequence, use the DROP statement.

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.

Related tasks:

- "Altering a sequence" on page 183

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Altering materialized query table properties

Procedure:

With some restrictions, you can change a materialized query table to a regular table or a regular table to a materialized query table. You cannot change other table types; only regular and materialized query tables can be changed. For example, you cannot change a replicated materialized query table to a regular table, nor the reverse.

Once a regular table has been altered to a materialized query table, the table is placed in a check pending state. When altering in this way, the fullselect in the materialized query 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 materialized query table is defined on an original table, then the original table cannot itself be altered into a materialized query table. If the original table has triggers, check constraints, referential constraints, or a defined unique index, then it cannot be altered into a materialized query table. If altering the table properties to define a materialized query 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 materialized query table, the fullselect of the materialized query table definition cannot reference the original table directly or indirectly through views, aliases, or materialized query tables.

To change a materialized query table to a regular table, use the following:

```
ALTER TABLE sumtable
   SET SUMMARY AS DEFINITION ONLY
```

To change a regular table to a materialized query table, use the following:

```
ALTER TABLE regtable
   SET SUMMARY AS <fullselect>
```

The restrictions on the fullselect when altering the regular table to a materialized query table are very much like the restrictions when creating a summary table using the CREATE SUMMARY TABLE statement.

Related tasks:

- "Creating a materialized query table" on page 121
- "Refreshing the data in a materialized query table" on page 185
- "Dropping a materialized query or staging table" on page 194

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Refreshing the data in a materialized query table

Procedure:

You can refresh the data in one or more materialized query 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 materialized query table:

```
REFRESH TABLE SUMTAB1
```

Related tasks:

- "Creating a materialized query table" on page 121
- "Altering materialized query table properties" on page 184

Related reference:

- "REFRESH TABLE statement" in the *SQL Reference, Volume 2*

Altering a user-defined structured type

Procedure:

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.

Related concepts:

- "User-defined structured types" in the *Application Development Guide: Programming Server Applications*
- "Structured type hierarchies" in the *Application Development Guide: Programming Server Applications*

Related tasks:

- "Creating structured types" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "ALTER TYPE (Structured) statement" in the *SQL Reference, Volume 2*

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.

Related concepts:

- "Typed tables" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "DELETE statement" in the *SQL Reference, Volume 2*
- "UPDATE statement" in the *SQL Reference, Volume 2*

Renaming an existing table or index

You can give an existing table or index a new name within a schema and maintain the authorizations and indexes that were created on the original table.

Prerequisites:

The existing table or index to be renamed can be an alias identifying a table or index.

Restrictions:

The existing table or index to be renamed must not be the name of a catalog table or index, a summary table or index, a typed table, a declared global temporary table, a nickname, or an object other than a table, a view, or an alias.

The existing table or index 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 or index being renamed is not affected by any of these restrictions.

Procedure:

To rename an existing table or index using the Control Center:

1. Expand the object tree until you see the **Tables** or **Views** folder.
2. Right-click on the table or view you want to rename, and select **Rename** from the pop-up menu.
3. Type the new table or view 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
```

To rename an existing index using the command line, enter:

```
RENAME INDEX <schema_name>.<index_name> TO <new_name>
```

The SQL statement below renames the EMPIND index within the COMPANY schema to MSTRIND:

```
RENAME INDEX COMPANY.EMPIND TO MSTRIND
```

Packages are invalidated and must be rebound if they refer to a table or index that has just been renamed. The packages are implicitly rebound regardless of whether another index exists with the same name. Unless a better choice exists, the package will use the same index it had before, under its new name.

Related reference:

- "RENAME statement" in the *SQL Reference, Volume 2*

Updating table and view contents using the MERGE statement

DB2 Universal Database provides the ability to update a table or a view using data from another source, typically the result of a table reference. This type of update is performed using the MERGE statement.

Rows in the target table that match the source can be deleted or updated based on specified directions from within the MERGE statement. Rows that do not exist in the target table can be inserted.

Updating, deleting, or inserting rows in a view cause corresponding row updates, deletions, or insertions in the table on which the view is based.

Restrictions:

The authorization ID associated with the MERGE statement must have the appropriate privileges to carry out any of the three possible actions: update, delete, or insert on the table or underlying table of the view. The authorization ID should also have the appropriate privileges on the table or underlying table of the view in the subquery.

If an error occurs in the MERGE statement, the entire set of operations associated with the MERGE is rolled back.

It is not possible to update a row in the target table or underlying table of the view that did not exist before the MERGE statement was run. That is, updating a row that was inserted as part of the MERGE statement is not allowed.

If a view is specified as the target of the MERGE statement, either no INSTEAD OF triggers should be defined for the view; or, an INSTEAD OF trigger should be defined for each of the update, delete, and insert operations.

Procedure:

To update, delete, insert, or perform any combination of these actions on a target table, enter the following at the command prompt:

```
MERGE INTO <table or view name>
   USING <table reference> ON <search condition>
   WHEN <match condition> THEN <modification operation or signal statement>
```

The modification operations and signal statements can be specified more than once per MERGE statement. Each row in the target table or view can be operated on only once within a single MERGE statement. This means that a row in the target table or view can be identified as MATCHED only with one row in the result table of the table reference.

Consider a situation where there are two tables: shipment and inventory. Using the shipment table, merge rows into the inventory table. For rows that match, increase the quantity in the inventory table by the quantity in the shipment table. Otherwise, insert the new part number into the inventory table.

```
MERGE INTO inventory AS in
   USING (SELECT partno, description, count FROM shipment
   WHERE shipment. partno IS NOT NULL) AS sh
   ON (in.partno = sh.partno)
   WHEN MATCHED THEN
      UPDATE SET
         description = sh.description
         quantity = in.quantity + sh.count
   WHEN NOT MATCHED THEN
      INSERT
         (partno, description, quantity)
         VALUES (sh.partno, sh.description, sh.count)
```

There is no DELETE option in this example. A more complex matching condition can allow for the addition of a DELETE option. There are several other options, such as the use of the signal statement and the ELSE clause, that are not documented here but can be found in the SQL Reference.

Related reference:

- "MERGE statement" in the *SQL Reference, Volume 2*

Dropping a table

Procedure:

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.
- 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.
- 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.

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related tasks:

- "Dropping a user-defined temporary table" on page 190

- "Recovering inoperative views" on page 194

Related reference:
- "DROP statement" in the *SQL Reference, Volume 2*

Dropping a user-defined temporary table

A user-defined temporary table is created using the DECLARE GLOBAL TEMPORARY TABLE statement.

Prerequisites:

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.

Restrictions:

Packages cannot be dependent on this type of table and therefore they are not invalidated when such a table is dropped.

Procedure:

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.

Related tasks:
- "Creating a user-defined temporary table" on page 96

Related reference:
- "DROP statement" in the *SQL Reference, Volume 2*
- "SET SCHEMA statement" in the *SQL Reference, Volume 2*

Dropping a trigger

Procedure:

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.

Related tasks:

- "Creating a trigger" on page 109

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Dropping a user-defined function (UDF), function mapping, or method

A user-defined function (UDF), function template, or function mapping can be dropped using the DROP statement.

Prerequisites:

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.

Restrictions:

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.

Procedure:

You can disable a function mapping with the mapping option DISABLE.

Packages which are marked inoperative are not implicitly rebound. The package must either be rebound using the BIND or REBIND commands or it must be prepared by use of the PREP command. 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.

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*
- "BIND Command" in the *Command Reference*
- "PRECOMPILE Command" in the *Command Reference*
- "REBIND Command" in the *Command Reference*

Dropping a user-defined type (UDT) or type mapping

You can drop a user-defined type (UDT) or type mapping using the DROP statement.

Restrictions:

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.

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.

Procedure:

The DROP statement is used to drop your user-defined type.

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.

Related concepts:

- "User-defined type (UDT)" on page 115

Related tasks:

- "Creating a user-defined distinct type" on page 116
- "Creating a type mapping" on page 118

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Altering or dropping a view

The ALTER VIEW statement modifies an existing view definition by altering a reference type column to add a scope. The DROP statement deletes a view.

Prerequisites:

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.

Restrictions:

Changes you make to the underlying content of a view require that you use triggers. Other changes to a view require that you drop and then re-create the view.

Procedure:

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). You can also modify the contents of a view through INSTEAD OF triggers.

Other database objects such as tables and indexes are not affected although packages and cached dynamic statements are marked invalid.

To alter the definition for 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.

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
```

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related tasks:

- "Creating a trigger" on page 109
- "Creating a view" on page 118
- "Recovering inoperative views" on page 194

Related reference:

- "ALTER VIEW statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*

Recovering inoperative views

Procedure:

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. (A superview is a typed view upon which another typed view, a subview, is based.)
- 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.

Related tasks:

- "Altering or dropping a view" on page 192

Related reference:

- "CREATE VIEW statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*
- "GRANT (Table, View, or Nickname Privileges) statement" in the *SQL Reference, Volume 2*
- "SYSCAT.VIEWS catalog view" in the *SQL Reference, Volume 1*

Dropping a materialized query or staging table

Procedure:

You cannot alter a materialized query or staging 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.

To drop a materialized query table using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click on the materialized query or staging table you want to drop, and select **Drop** from the pop-up menu.
3. Check the **Confirmation** box, and click **Ok**.

To drop a materialized query or staging table using the command line, enter:

```
DROP TABLE <table_name>
```

The following SQL statement drops the materialized query table XT:

```
DROP TABLE XT
```

A materialized query table may be explicitly dropped with the DROP TABLE statement, or it may be dropped implicitly if any of the underlying tables are dropped.

A staging table may be explicitly dropped with the DROP TABLE statement, or it may be dropped implicitly when its associated materialized query table is dropped.

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related tasks:

- "Creating a materialized query table" on page 121
- "Creating a staging table" on page 126

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*

Recovering inoperative summary tables

Procedure:

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.

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*
- "GRANT (Table, View, or Nickname Privileges) statement" in the *SQL Reference, Volume 2*
- "SYSCAT.VIEWS catalog view" in the *SQL Reference, Volume 1*

Dropping an index, index extension, or an index specification

Restrictions:

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.)

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.

Procedure:

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
```

Any packages and cached dynamic SQL statements that depend on the dropped indexes are marked invalid. The application program is not affected by changes resulting from adding or dropping indexes.

Related concepts:

- "Statement dependencies when changing objects" on page 197

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "DROP statement" in the *SQL Reference, Volume 2*

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.

Packages and cached dynamic SQL statements can be dependent on many types of 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.

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

Related concepts:

- "Package Creation Using the BIND Command" in the *Application Development Guide: Programming Client Applications*
- "Application, Bind File, and Package Relationships" in the *Application Development Guide: Programming Client Applications*
- "Package Rebinding" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "DROP statement" in the *SQL Reference, Volume 2*
- "SYSCAT.PACKAGEAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.PACKAGEDEP catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.PACKAGES catalog view" in the *SQL Reference, Volume 1*
- "BIND Command" in the *Command Reference*
- "REBIND Command" in the *Command Reference*

Part 2. Database Security

Chapter 7. 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 8, "Auditing DB2 Universal Database™ (DB2 UDB) activities," on page 251.

The following topics are discussed:

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.

Security issues when installing DB2 Universal Database

Security issues are important to the DB2® administrator from the moment the product is installed.

To complete the installation of DB2 Universal Database™ (DB2 UDB), a user ID, a group name, and a password are required. The GUI-based DB2 UDB install program creates default values for different user IDs and the group. Different defaults are created, depending on whether you are installing on UNIX® or Windows® platforms:

- On UNIX platforms, the DB2 UDB install program creates different default users for the DAS (`dasusr`), the instance owner (`db2inst`), and the fenced user (`db2fenc`).

  The DB2 UDB install program appends a number from 1-99 to the default user name, until a user ID that does not already exist can be created. For example, if the users `db2inst1` and `db2inst2` already exist, the DB2 UDB install program creates the user `db2inst3`. If a number greater than 10 is used, the character portion of the name is truncated in the default user ID. For example, if the user

ID db2fenc9 already exists, the DB2 UDB install program truncates the c in the user ID, then appends the 10 (db2fen10). Truncation does not occur when the numeric value is appended to the default DAS user (for example, dasusr24).

- On Windows platforms, the DB2 UDB install program creates the default user db2admin for the DAS user, the instance owner, and fenced users. Unlike UNIX platforms, no numeric value is appended to the user ID.

To minimize the risk of a user other than the administrator from learning of the defaults and using them in an improper fashion within databases and instances, change the defaults during the install to a new or existing user ID of your choice.

Note: Response file installations do not use default values for user IDs or group names. These values must be specified in the response file.

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, users will be allowed to connect. For example on UNIX operating systems, undefined passwords are treated as NULL. In this situation, any user without a defined password will be considered to 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. Use passwords at the operating system level if you want the operating system to do the authentication of users for your database.

Note: You cannot use undefined passwords if you want your database environment to adhere to Common Criteria requirements.

After installing DB2 Universal Database also review, and change (if required), the default privileges that have been granted to users. By default, the installation process grants system administration (SYSADM) privileges to the following users on each operating system:

Windows 9x Any Windows 98, or Windows ME user.

Other Windows environments
On Windows NT®, Windows 2000, Windows XP, or Windows Server 2003, a valid DB2 UDB username that belongs to the Administrators group.

UNIX platforms A valid DB2 UDB username that belongs to the primary group of the instance owner.

SYSADM privileges are the most powerful set of privileges available within DB2 Universal Database. As a result, you may not want all of these users to have SYSADM privileges by default. DB2 UDB provides the administrator with the ability to grant and revoke privileges to groups and individual user IDs.

By updating the database manager configuration parameter *sysadm_group*, the administrator can control which group of users possesses SYSADM privileges. You must follow the guidelines below to complete the security requirements for both DB2 UDB 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.

The administrator 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 use this instance-owner user ID only 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 can modify the instance, or any object within the instance.

The created user ID must be associated with a password to provide authentication before being permitted entry into the data and databases within the instance. The recommendation when creating a password is to follow your organization's password naming guidelines.

Related concepts:

- "Naming rules in an NLS environment" on page 297
- "Naming rules in a Unicode environment" on page 297
- "Windows NT platform security considerations for users" on page 205
- "UNIX platform security considerations for users" on page 206
- "Authentication" in the *Administration Guide: Planning*
- "Authorization" in the *Administration Guide: Planning*
- "Location of the instance directory" on page 207
- "General naming rules" on page 291
- "User, user ID and group naming rules" on page 294

Acquiring Windows users' group information using an access token

An access token is an object that describes the security context of a process or thread. The information in an access token includes the identity and privileges of the user account associated with the process or thread.

When you log on, the system verifies your password by comparing it with information stored in a security database. If the password is authenticated, the system produces an access token. Every process run on your behalf uses a copy of this access token.

An access token can also be acquired based on cached credentials. Once you have been authenticated to the system, your credentials are cached by the operating system. The access token of the last logon can be referenced in the cache when it is not possible to contact the domain controller.

The access token includes information about all of the groups you belong to: local groups and various domain groups (global groups, domain local groups, and universal groups).

Note: Group lookup using client authentication is not supported using a remote connection even though access token support is enabled.

To enable access token support, you must use the **db2set** command to update the DB2®_GRP_LOOKUP registry variable. Your choices when updating this registry variable include:

- TOKEN

This choice enables access token support to lookup all groups that the user belongs to at the location where the user account is defined. This location is typically either at the domain or local to the DB2 Universal Database™ (DB2 UDB) server.

- TOKENLOCAL

  This choice enables access token support to lookup all local groups that the user belongs to on the DB2 UDB server.

- TOKENDOMAIN

  This choice enables access token support to lookup all domain groups that the user belongs to on the domain.

When enabling access token support, there are several limitations that affect your account management infrastructure. When this support is enabled, DB2 UDB collects group information about the user who is connecting to the database. Subsequent operations after a successful CONNECT or ATTACH request that have dependencies on other authorization IDs will still need to use conventional group enumeration. The access token advantages of nested global groups, domain local groups, and cached credentials will not be available. For example, if, after a connection, the SET SESSION_USER is used to run under another authorization ID, only the conventional group enumeration is used to check what rights are given to the new authorization ID for the session. You will still need to grant and revoke explicit privileges to individual authorization IDs known to DB2 UDB, as opposed to the granting and revoking of privileges to groups to which the authorization IDs belongs.

If you intend to assign groups to SYSADM, SYSMAINT, or SYSCTRL, you need to ensure that the assigned groups are not nested global groups, nor domain local groups, and then the cached credential capability is not needed.

You should consider using the DB2_GRP_LOOKUP registry variable and specify the group lookup location to indicate where DB2 UDB should lookup groups using the conventional group enumeration methodology. For example,

```
db2set DB2_GRP_LOOKUP=LOCAL,TOKENLOCAL
```

This enables the access token support for enumerating local groups. Group lookup for an authorization ID different from the connected user is performed at the DB2 UDB server.

```
db2set DB2_GRP_LOOKUP=,TOKEN
```

This enables the access token support for enumerating groups at the location where the user ID is defined. Group lookup for an authorization ID different from the connected user is performed where the user ID is defined.

```
db2set DB2_GRP_LOOKUP=DOMAIN,TOKENDOMAIN
```

This enables the access token support for enumerating domain groups. Group lookup for an authorization ID different from the connected user is performed where the user ID is defined.

Applications using dynamic SQL in a package bound using DYNAMICRULES RUN (which is the default) is run under the privileges of the person who runs the application. In this case, the already mentioned limitations do not apply. This would include applications written to use JDBC and DB2 CLI.

Access token support can be enabled with all authentications types except CLIENT authentication.

Note: For Windows® NT 4.0 users, the access token support only supports the security context at the process level if your DB2 UDB applications are using local implicit connections. That is, all threads in the applications are treated as if they were running under the security context of the user executing the applications. If you require different user security contexts for different threads, you should consider moving to Windows 2000 or later; or consider changing your DB2 UDB applications to use explicit connections.

Related concepts:

- "Security issues when installing DB2 Universal Database" on page 201

Details on security based on operating system

Each operating system provides ways to manage security. Some of the security issues associated with the operating systems are discussed in this section.

Note: The cryptographic routines DB2 Universal Database™ (DB2 UDB) uses to perform encryption of the userid and password when using SERVER_ENCRYPT authentication, and of the userid, password and user data when using DATA_ENCRYPT authentication are compliant with FIPS 140-2.

The cryptographic routines are provided by IBM Crypto for C (ICC) Version 1.2.1. The FIPS 140-2 Validation Certificate No. 384 for ICC can be found on the NIST website at: http://csrc.nist.gov/cryptval/140-1/140crt/140crt384.pdf

The Security Policy can also be found on the NIST website at: http://csrc.nist.gov/cryptval/140-1/140sp/140sp384.pdf

The Security Policy has guidance that must be followed in order to install DB2 UDB in a FIPS 140-2 compliant manner, and section 5.3.2 should be consulted for further information.

The FIPS 140-2 compliance is only available on the following systems: AIX, Microsoft Windows, Solaris operating environment, Linux, and HP-UX. The Validation Certificate and Security Policy should be consulted for the full details of supported systems.

Windows NT platform security considerations for users

System Administration (SYSADM) authority is granted to any valid DB2® Universal Database (DB2 UDB) user account which belongs to the local Administrators group on the machine where the account is defined.

By default in a Windows® domain environment, only domain users that belong to the Administrators group at the Domain Controller have SYSADM authority on an instance. Since DB2 UDB 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.

Note: In a domain environment such as is found in Windows NT®, DB2 UDB only authenticates the first 64 groups that meet the requirements and restrictions, and to which a user ID belongs. You may have more than 64 groups.

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
```

Related concepts:

- "UNIX platform security considerations for users" on page 206

Windows local system account support

On Windows® platforms, DB2® Universal Database (DB2 UDB) supports applications running under the context of the local system account (LSA) with local implicit connection. Developers writing applications to be run under this account need to be aware that DB2 UDB has restrictions on objects with schema names starting with "SYS". Therefore if your applications contain DDLs that create DB2 UDB objects, they should be written such that:

- For static SQL, they should be bounded with a value for the QUALIFIER options other than the default one.
- For dynamic SQL, the objects to be created should be explicitly qualified with a schema name supported by DB2 UDB, or the CURRENT SCHEMA register must be set to a schema name supported by DB2 UDB.

Group information for the LSA is gathered at the first group lookup request after the DB2 UDB instance is started and will not be refreshed until the instance is restarted.

Related concepts:

- "Security issues when installing DB2 Universal Database" on page 201

UNIX platform security considerations for users

DB2® Universal Database (DB2 UDB) does not support root acting directly as a database administrator. You should use **su - <instance owner>** as the database administrator.

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.

Related concepts:

- "Windows NT platform security considerations for users" on page 205

Location of the instance directory

On UNIX®, the **db2icrt** command creates the main SQL library (`sqllib`) directory under the home directory of the instance owner.

On Windows® operating systems, the instance directory is located in the `/sqllib` sub-directory, in the directory where DB2® was installed.

Related concepts:

- "Instance creation" on page 15

Related tasks:

- "Creating additional instances" on page 20

Security plug-ins

Authentication in DB2® Universal Database (DB2 UDB) is done through *security plug-ins*. For more information, see "Security plug-ins" from the *Application Development Guide: Programming Client Applications*.

Authentication methods 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. 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.

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.

Notes:

1. 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.
2. If you are installing DB2® Universal Database (DB2 UDB) to set up a Common Criteria certified configuation, you must specify SERVER.

SERVER_ENCRYPT
Specifies that the server accepts encrypted SERVER authentication schemes. If the client authentication is not specified, the client is authenticated using the method selected at the server.

If the client authentication is SERVER, the client is authenticated by passing the user ID and password to the server. If the client authentication is SERVER_ENCRYPT, the client is authenticated by passing an encrypted user ID and encrypted password.

If SERVER_ENCRYPT is specified at the client and SERVER is specified at the server, an error is returned because of the mismatch in the authentication levels.

CLIENT

Specifies that authentication occurs on the database partition where the application is invoked using operating system security. The user ID and password specified during a connection or attachment attempt are compared with the valid user ID and password combinations on the client node to determine if the user ID is permitted access to the instance. No further authentication will take place on the database server. This is sometimes called single signon.

If the user performs a local or client login, the user is known only to that local client workstation.

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® 9x 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.

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 or USING clauses.

To protect against all clients except DRDA® clients from DB2 for OS/390® and z/OS™, DB2 for VM and VSE, and DB2 for iSeries™, 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 5. 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

KERBEROS

Used when both the DB2 UDB 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 UDB server. The KERBEROS authentication type is supported on clients and servers running Windows 2000, AIX®, and Solaris operating environment.

Kerberos authentication works as follows:

1. A user logging on to the client machine using a domain account authenticates to the Kerberos key distribution center (KDC) at the domain controller. The key distribution center issues a ticket-granting ticket (TGT) to the client.
2. During the first phase of the connection the server sends the target principal name, which is the service account name for the DB2 UDB server service, to the client. Using the server's target principal name and the target-granting ticket, the client requests a service ticket from the ticket-granting service (TGS) which also resides at the domain controller. If both the client's ticket-granting ticket and the server's target principal name are valid, the TGS issues a service ticket to the client. The principal name recorded in the database directory may now be specified as name/instance@REALM. (This is in addition to the current DOMAIN\userID and userID@xxx.xxx.xxx.com formats accepted on Windows 2000 with DB2 UDB Version 7.1 and following.)

3. The client sends this service ticket to the server via the communication channel (which may be, as an example, TCP/IP).
4. The server validates the client's server ticket. If the client's service ticket is valid, then the authentication is completed.

It is possible to catalog the databases on the client machine and explicitly specify the Kerberos authentication type with the server's target principal name. In this way, the first phase of the connection can be bypassed.

If a user ID and a password are specified, the client will request the ticket-granting ticket for that user account and use it for authentication.

KRB_SERVER_ENCRYPT
Specifies that the server accepts KERBEROS authentication or encrypted SERVER authentication schemes. If the client authentication is KERBEROS, the client is authenticated using the Kerberos security system. If the client authentication is SERVER_ENCRYPT, the client is authenticated using a user ID and encryption password. If the client authentication is not specified, then the client will use Kerberos if available, otherwise it will use password encryption. For other client authentication types, an authentication error is returned. The authenticaion type of the client cannot be specified as KRB_SERVER_ENCRYPT

Note: The Kerberos authentication types are only supported on clients and servers running Windows 2000, Windows XP, Windows Windows Server 2003, and AIX operating systems, as well as Solaris operating environment. Also, both client and server machines must either belong to the same Windows domain or belong to trusted domains. This authentication type should be used when the server supports Kerberos and some, but not all, of the client machines support Kerberos authentication.

DATA_ENCRYPT
The server accepts encrypted SERVER authentication schemes and the encryption of user data. The authentication works exactly the same way as that shown with SERVER_ENCRYPT. See that authentication type for more information.

The following user data are encrypted when using this authentication type:

- SQL statements.
- SQL program variable data.
- Output data from the server processing of an SQL statement and including a description of the data.
- Some or all of the answer set data resulting from a query.
- Large object (LOB) data streaming.
- SQLDA descriptors.

DATA_ENCRYPT_CMP
The server accepts encrypted SERVER authentication schemes and the encryption of user data. In addition, this authentication type allows compatibility with down level products not supporting DATA_ENCRYPT authentication type. These products are permitted to connect with the SERVER_ENCRYPT authentication type and without encrypting user data. Products supporting the new authentication type must use it. This authentication type is only valid in the server's database manager configuration file and is not valid when used on on the CATALOG DATABASE command.

GSSPLUGIN
Specifies that the server uses a GSS-API plug-in to perform authentication. If the client authentication is not specified, the server returns a list of server-supported plug-ins, including any Kerberos plug-in that is listed in the *srvcon_gssplugin_list* database manager configuration parameter, to the client. The client selects the first plug-in found in the client plug-in directory from the list. If the client does not support any plug-in in the list, the client is authenticated using the Kerberos authentication scheme (if it is returned). If the client authentication is the GSSPLUGIN authentication scheme, the client is authenticated using the first supported plug-in in the list.

GSS_SERVER_ENCRYPT
Specifies that the server accepts plug-in authentication or encrypted server authentication schemes. If client authentication occurs via a plug-in, the client is authenticated using the first client-supported plug-in in the list of server-supported plug-ins.

If the client authentication is not specified and an implicit connect is being performed (that is, the client does not supply a user ID and password when making the connection), the server returns a list of server-supported plug-ins, the Kerberos authentication scheme (if one of the plug-ins in the list is Kerberos-based), and the encrypted server authentication scheme. The client is authenticated using the first supported plug-in found in the client plug-in directory. If the client does not support any of the plug-ins that are in the list, the client is authenticated using the Kerberos authentication scheme. If the client does not support the Kerberos authentication scheme, the client is authenticated using the encrypted server authentication scheme, and the connection will fail because of a missing password. A client supports the Kerberos authentication scheme if a DB2 UDB-supplied Kerberos plug-in exists for the operating system, or a Kerberos-based plug-in is specified for the *srvcon_gssplugin_list* database manager configuration parameter.

If the client authentication is not specified and an explicit connection is being performed (that is, both the user ID and password are supplied), the authentication type is equivalent to SERVER_ENCRYPT.

Notes:

1. 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 UDB system, you have a fail-safe option available on all platforms that will allow you to override the usual DB2 UDB 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 UDB command. This special user is identified as follows:

- UNIX® platforms: the instance owner
- NT platform: someone belonging to the local "administrators" group
- Other platforms: there is no local security on the other platforms, so all users pass local security checks anyway

Related concepts:

- "Authentication considerations for remote clients" on page 212
- "Partitioned database authentication considerations" on page 212
- "DB2 for Windows NT and Windows NT security introduction" on page 361

Related reference:

- "authentication - Authentication type configuration parameter" in the *Administration Guide: Performance*
- "trust_allclnts - Trust all clients configuration parameter" in the *Administration Guide: Performance*
- "trust_clntauth - Trusted clients authentication configuration parameter" in the *Administration Guide: Performance*

Authentication considerations for remote clients

When cataloging a database for remote access, the authentication type can be specified in the database directory entry.

For databases accessed using DB2® Connect: If a value is not specified, SERVER authentication is assumed.

The authentication type is not required. If it is not specified, the client will default to SERVER_ENCRYPT. However, if the server does not support SERVER_ENCRYPT, the client attempts to retry using a value supported by the server. If the server supports multiple authentication types, the client will not choose among them, but instead returns an error. The error is returned to ensure that the correct authentication type is used. In this case, the client must catalog the database using a supported authentication type. If an authentication type is specified, authentication can begin immediately provided that value specified matches that at the server. If a mismatch is detected, DB2 Universal Database™ (DB2 UDB) attempts to recover. Recovery may result in more flows to reconcile the difference, or in an error if DB2 UDB cannot recover. In the case of a mismatch, the value at the server is assumed to be correct.

Related concepts:

- "Authentication methods for your server" on page 207

Partitioned database authentication 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.

Related concepts:

- "Authentication methods for your server" on page 207

Kerberos authentication details

DB2® Universal Database (DB2 UDB) provides support for the Kerberos authentication protocol on AIX® Version 5.2, Solaris operating environment Version 8, Red Hat Enterprise Linux Server Advanced Server 2.1 (32-bit Intel), and Windows® 2000 and later operating systems.

The Kerberos support is provided as a GSS-API security plugin named "IBMkrb5" which is used as both a server and as a client authentication plugin. The library is placed in the `sqllib/security{32|64}/plugin/IBM/{client|server}` directories for UNIX® and Linux; and the `sqllib/security/plugin/IBM{client|server}` directories for Windows.

Note: For 64-bit Windows, the plugin library is called `IBMkrb564.dll`. Furthermore, the actual plugin source code for the UNIX and Linux plugin, `IBMkrb5.C`, is available in the `sqllib/samples/security/plugins` directory.

A good understanding of using and configuring Kerberos is strongly recommended before attempting to use Kerberos authentication with DB2 UDB.

Kerberos description and Introduction

Kerberos is a third party network authentication protocol that employs a system of shared secret keys to securely authenticate a user in an unsecured network environment. First developed at MIT in the late 1980s, the latest revision, Kerberos 5, is described by Internet Engineering Task Force (IETF) RFC 1510. (The URL for IETF is http://www.ieft.org. RFC 1510 is titled "The Kerberos Network Authentication Service (V5)".) A three-tiered system is used in which encrypted tickets (provided by a separate server called the Kerberos Key Distribution Center, or KDC for short) are exchanged between the application server and client rather than a text user ID and password pair. These encrypted service tickets (called *credentials*) have a finite lifetime and are only understood by the client and the server. This reduces the security risk, even if the ticket is intercepted from the network. Each user, or *principal* in Kerberos terms, possesses a private encryption key that is shared with the KDC. Collectively, the set of principals and computers registered with a KDC are known as a *realm*.

A key feature of Kerberos is that it permits a single sign-on environment whereby a user only needs to verify his identity to the resources within the Kerberos realm once. When working with DB2 UDB, this means that a user is able to connect or attach to a DB2 UDB server without providing a user ID or password. Another advantage is that the user ID administration is simplified because a central repository for principals is used. Finally, Kerberos supports mutual authentication which allows the client to validate the identity of the server.

Kerberos set-up

DB2 UDB and its support of Kerberos relies upon the Kerberos layer being installed and configured properly on all machines involved prior to the involvement of DB2 UDB. This includes, but is not necessarily limited to, the following requirements:

1. The client and server machines and principals must belong to the same realm, or else trusted realms (or trusted domains in the Windows terminology)

2. Creation of appropriate principals
3. Creation of server keytab files, where appropriate
4. All machines involved must have their system clocks synchronized (Kerberos typically permits a 5 minute time skew, otherwise a preauthentication error may occur when obtaining credentials).

For details on installing and configuring Kerberos please refer to the documentation provided with the installed Kerberos product.

The sole concern of DB2 UDB will be whether the Kerberos security context is successfully created based on the credentials provided by the connecting application (that is, authentication). Other Kerberos features, such as the signing or encryption of messages, will not be used. Furthermore, whenever available, mutual authentication will be supported.

The Kerberos prerequisites are as follows:

- AIX Version 5.2 with IBM® Network Authentication Service (NAS) Toolkit 1.3
- Solaris operating environment Version 8 with SEAM (Sun Enterprise Authentication Mechanism) and IBM NAS Toolkit 1.3
- Red Hat Enterprise Linux Advanced Server 2.1 with the `krb5-libs` and `krb5-workstation` filesets
- Windows 2000 (and later operating systems) Server

Kerberos and client principals

The principal may be found in either a 2-part or multi-part format, (that is, *name@REALM* or *name/instance@REALM*). As the "name" part will be used in the authorization ID (AUTHID) mapping, the name must adhere to the DB2 UDB naming rules. This means that the name may be up to 30 characters long and it must adhere to the existing restrictions on the choice of characters used. (AUTHID mapping is discussed in a later topic.)

Note: Windows directly associates a Kerberos principal with a domain user. An implication of this is that Kerberos authentication is not available to Windows machines that are not associated with a domain or realm. Furthermore, Windows only supports 2-part names (that is, *name@domain*).

The principal itself must be capable of obtaining outbound credentials with which it may request and receive service tickets to the target database. This is normally accomplished with the **kinit** command on UNIX or Linux, and is done implicitly at logon time on Windows.

Kerberos and authorization ID mapping

Unlike operating system user IDs whose scope of existence is normally restricted to a single machine (NIS being a notable exception), Kerberos principals have the ability to be authenticated in realms other than their own. The potential problem of duplicated principal names is avoided by using the realm name to fully qualify the principal. In Kerberos, a fully qualified principal takes the form `name/instance@REALM` where the instance field may actually be multiple instances separated by a "/", that is, `name/instance1/instance2@REALM`, or it may be omitted altogether. The obvious restriction is that the realm name must be unique within all the realms defined within a network. The problem for DB2 UDB is that in order to provide a simple mapping from the principal to the AUTHID, a one-to-one mapping between the principal name, that is, the "name" in the fully qualified

principal, and the AUTHID is desirable. A simple mapping is needed as the AUTHID is used as the default schema in DB2 UDB and should be easily and logically derived. As a result, the database administrator needs to be aware of the following potential problems:

- Principals from different realms but with the same name will be mapped to the same AUTHID.
- Principals with the same name but different instances will be mapped to the same AUTHID.

Giving consideration to the above, the following recommendations are made:

- Maintain an unique namespace for the name within all the trusted realms that will access the DB2 UDB server
- All principals with the same name, regardless of the instance, should belong to the same user.

Kerberos and server principals

On UNIX or Linux, the server principal name for the DB2 UDB instance is assumed to be `<instance name>/<fully qualified hostname>@REALM`. This principal must be able to accept Kerberos security contexts and it must exist before starting the DB2 UDB instance since the server name is reported to DB2 UDB by the plugin at initialization time.

On Windows, the server principal is taken to be the domain account under which the DB2 UDB service started. An exception to this is the instance may be started by the local SYSTEM account, in which case, the server principal name is reported as `host/<hostname>`; this is only valid if both the client and server belong to Windows domains.

Windows does not support greater than 2-part names. This poses a problem when a Windows client attempts to connect to a UNIX server. As a result, a Kerberos principal to Windows account mapping may need to be set up in the Windows domain if interoperability with UNIX Kerberos is required. (Please refer to the appropriate Microsoft® documentation for relevant instructions.)

Kerberos keytab files

Every Kerberos service on UNIX or Linux wishing the accept security context requests must place its credentials in a *keytab* file. This applies to the principals used by DB2 UDB as server principals. Only the default keytab file is searched for the server's key. For instructions on adding a key to the keytab file, please refer to the documentation provided with the Kerberos product.

There is no concept of a keytab file on Windows and the system automatically handles storing and acquiring the credentials handle for a principal.

Kerberos and groups

Kerberos is an authentication protocol that does not possess the concept of groups. As a result DB2 UDB relies upon the local operating system to obtain a group list for the Kerberos principal. For UNIX or Linux, this requires that an equivalent system account should exist for each principal. For example, for the principal `name@REALM`, DB2 UDB collects group information by querying the local operating system for all group names to which the operating system user *name* belongs. If an operating system user does not exist, then the AUTHID will only belong to the

PUBLIC group. Windows, on the other hand, automatically associates a domain account to a Kerberos principal and the additional step to create a separate operating system account is not required.

Enabling Kerberos authentication on the client

The *clnt_krb_plugin* database manager configuration parameter should be updated to the name of the Kerberos plugin being used. On the supported platforms this should be set to `IBMkrb5`. This parameter will inform DB2 UDB that it is capable of using Kerberos for connections and local instance-level actions if the AUTHENTICATION parameter is set to KERBEROS or KRB_SERVER_ENCRYPT. Otherwise, no client-side Kerberos support is assumed.

Note: No checks are performed to validate that Kerberos support is available.

Optionally, when cataloging a database on the client, an authentication type may be specified:

```
db2 catalog db testdb at node testnode authentication kerberos target
   principal service/host@REALM
```

However, if the authentication information is not provided, then the server sends the client the name of the server principal.

Enabling Kerberos authentication on the server

The *srvcon_gssplugin_list* database manager configuration parameter should be updated with the server Kerberos plugin name. Although this parameter may contain a list of supported plugins, only one Kerberos plugin may be specified. However, if this field is blank and AUTHENTICATION is set to KERBEROS or KRB_SERVER_ENCRYPT, the default Kerberos plugin (`IBMkrb5`) is assumed and used. Either the AUTHENTICATION or SVRCON_AUTH parameter should be set to KERBEROS or KRB_SERVER_ENCRYPT if Kerberos authentication is to be used depending upon whether it is used for everything or just for incoming connections.

Creating a Kerberos plugin

There are several considerations you should consider when creating a Kerberos plugin:

- Write a Kerberos plugin as a GSS-API plugin with the notable exception that the *plugintype* in the function pointer array returned to DB2 UDB in the initialization function must be set to DB2SEC_PLUGIN_TYPE_KERBEROS.
- Under certain conditions, the server principal name may be reported to the client by the server. As such, the principal name should not be specified in the GSS_C_NT_HOSTBASED_SERVICE format (`service@host`), since DRDA® stipulates that the principal name be in the GSS_C_NT_USER_NAME format (`server/host@REALM`).
- In a typical situation, the default keytab file may be specified by the KRB5_KTNAME environment variable. However, as the server plugin will run within a DB2 UDB engine process, this environment variable may not be accessible.

Related concepts:

- "Authentication methods for your server" on page 207

Privileges, authority levels, and database authorities

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. *Database authorities* enable users to perform activities at the database level. Privileges, authority levels, and database authorities can be used together to control access to the database manager and its database objects. Users can access only those objects for which they have the required privilege, authority level, or database authority, which DB2® Universal Database (DB2 UDB) determines when it performs an authorization check for an authenticated user.

The database manager requires that each user be specifically authorized, either implicitly or explicitly, to use each database function needed to perform a specific task. *Explicit* authorities or privileges are granted to the user (GRANTEETYPE of U in the database catalogs). *Implicit* authorities or privileges are granted to a group to which the user belongs (GRANTEETYPE of G in the database catalogs). Thus, to create a table, a user must be authorized to create tables; to alter a table, a user must be authorized to alter the table, and so on.

Figure 3 illustrates the relationship between authorities and their span of control (database, database manager).

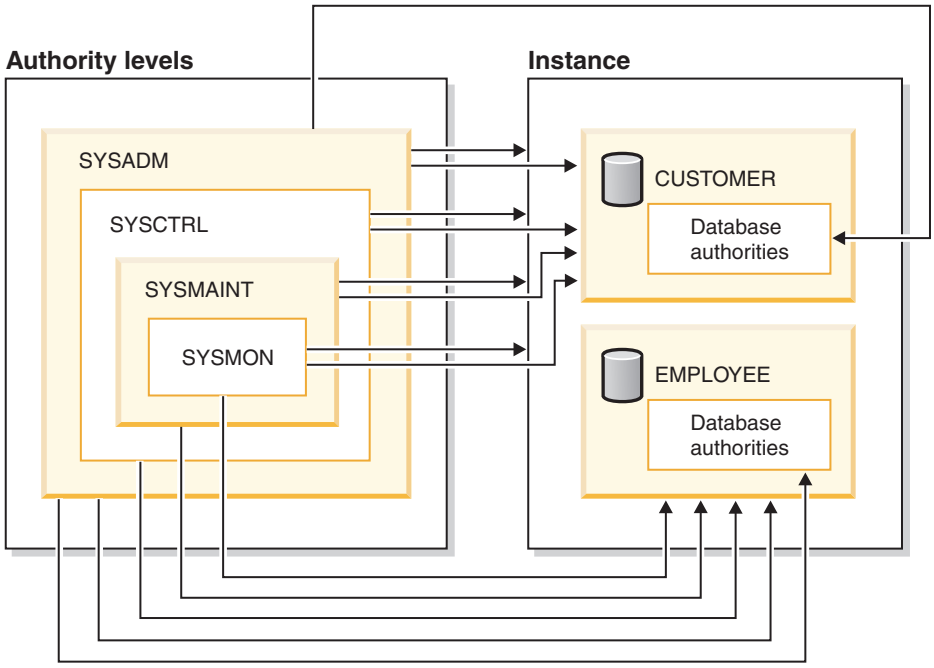


Figure 3. Hierarchy of Authorities

A user or group can have one or more of the following authorities or privileges:

- Administrative authority:
  - SYSADM (system administrator)

    The SYSADM authority level provides control over all the resources created and maintained by the database manager. The system administrator possesses all the authorities of DBADM, SYSCTRL, SYSMAINT, and SYSMON, and the authority to grant and revoke DBADM authority.

The user who possesses SYSADM authority is responsible both for controlling the database manager, and for ensuring the safety and integrity of the data. SYSADM authority provides implicit privileges on all objects in the database, control over which users can access the database manager, and the extent of this access. For more information about SYSADM authority, see ″System administration authority (SYSADM)″.

  - DBADM (database administrator)

    The DBADM database authority provides administrative authority over a single database. This database administrator possesses the privileges required to create objects, issue database commands, and access table data. The database administrator can also grant and revoke CONTROL and individual privileges. For more information about DBADM authority, see ″Database administration authority (DBADM)″.

- System control authority:
  - SYSCTRL (system control)

    The SYSCTRL authority level provides control over operations that affect system resources. For example, a user with SYSCTRL authority can create, update, stop, or drop a database. This user can also stop an instance, but cannot access table data. Users with SYSCTRL authority also have SYSMON authority. For more information about SYSCTRL authority, see ″System control authority (SYSCTRL)″.

  - SYSMAINT (system maintenance)

    The SYSMAINT authority level provides the authority required to perform maintenance operations on all databases associated with an instance. A user with SYSMAINT authority can update the database configuration, backup a database or table space, restore an existing database, and monitor a database. Like SYSCTRL, SYSMAINT does not provide access to table data. Users with SYSMAINT authority also have SYSMON authority. For more information about SYSMAINT authority, see ″System maintenance authority (SYSMAINT)″.

- SYSMON (system monitor authority)

  The SYSMON authority level provides the authority required to use the database system monitor. For more information about SYSMON authority, see ″System monitor authority (SYSMON)″

- Database authorities

  To perform activities such as creating a table or a routine, or for loading data into a table, specific database authorities are required. For more information, see ″Database authorities″.

- Privileges:

  Privileges are required to perform activities on database objects (for example, to create and drop an index). Privileges strictly define the tasks that a user can perform. For example, a user may have the privilege to create an index on a table, but not a trigger on the same table.

  - CONTROL privilege

    Possessing the CONTROL privilege on an object allows a user to access that database object, and to grant and revoke privileges to or from other users on that object.

    Note: The CONTROL privilege only apples to tables, views, nicknames, indexes, and packages.

If a different user requires the CONTROL privilege to that object, a user with SYSADM or DBADM authority must grant the CONTROL privilege to that object. The CONTROL privilege cannot be revoked from the object owner.

In some situations, the creator of an object automatically obtains the CONTROL privilege on that object. For more information, see ″Object creation, ownership, and privileges″.

- Individual privileges can be granted to allow a user to carry out specific tasks on specific objects.

  Users with administrative authority (SYSADM or DBADM) or the CONTROL privilege can grant and revoke privileges to and from users.

  Individual privileges and database authorities allow a specific function, but do not include the right to grant the same privileges or authorities to other users. The right to grant table, view, schema, package, routine, and sequence privileges to others can be extended to other users through the WITH GRANT OPTION on the GRANT statement. 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 the CONTROL privilege to revoke the privilege.

  Privileges can also be granted to PUBLIC. PUBLIC privileges apply to all users (authorization names), including any future users, regardless of whether any individual users have previously been granted the privilege.

- 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.

A user or group can be authorized for any combination of individual privileges or authorities. When a privilege is associated with an object, that object 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.

The REVOKE statement is used to revoke previously granted privileges. In DB2 UDB, the revoking of a privilege from an authorization name revokes the privilege granted by all authorization names.

Revoking a privilege from an authorization name does not revoke that same privilege from any other authorization names that were granted the privilege by that authorization name. For example, assume that CLAIRE grants SELECT WITH GRANT OPTION to RICK, then RICK grants SELECT to BOBBY and CHRIS. If CLAIRE revokes the SELECT privilege from RICK, BOBBY and CHRIS still retain the select privilege.

Related concepts:

- “System administration authority (SYSADM)” on page 221
- “System control authority (SYSCTRL)” on page 222
- “System maintenance authority (SYSMAINT)” on page 222
- “Database administration authority (DBADM)” on page 223
- “LOAD authority” on page 225

Object creation, ownership, and privileges

When an object is created, one authorization name is assigned *ownership* of the object. Ownership means that the user is authorized to reference the object in any SQL statement.

When an object is created within a schema, the authorization ID of the statement must have the required privilege to create objects in the implicitly or explicitly specified schema. That is, the authorization name must either be the owner of the schema, or possess the CREATEIN privilege on the schema.

Note: This requirement is not applicable when creating table spaces, buffer pools or database partition groups. These objects are not created in schemas.

When an object is created, the authorization ID of the statement is the owner of that object.

Note: One exception exists. If the AUTHORIZATION option is specified for the CREATE SCHEMA statement, any other object that is created as part of the CREATE SCHEMA operation is owned by the authorization ID specified by the AUTHORIZATION option. Any objects that are created in the schema after the initial CREATE SCHEMA operation, however, are owned by the authorization ID associated with the specific CREATE statement.

For example, the statement `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C! INT)` creates the schema `SCOTTSTUFF` and the table `SCOTTSTUFF.T1`, which are both owned by SCOTT. Assume that the user BOBBY is granted the CREATEIN privilege on the `SCOTTSTUFF` schema and creates an index on the `SCOTTSTUFF.T1` table. Because the index is created after the schema, BOBBY owns the index on `SCOTTSTUFF.T1`.

Privileges are assigned to the object owner based on the type of object being created:

- The CONTROL privilege is implicitly granted on newly created tables, indexes, and packages. This privilege allows the object creator to access the database object, and to grant and revoke privileges to or from other users on that object. If a different user requires the CONTROL privilege to that object, a user with SYSADM or DBADM authority must grant the CONTROL privilege to that object. The CONTROL privilege cannot be revoked by the object owner.

- The CONTROL privilege is implicitly granted on newly created views if the object owner has the CONTROL privilege on all the tables, views, and nicknames referenced by the view definition.
- Other objects like triggers, routines, sequences, table spaces, and buffer pools do not have a CONTROL privilege associated with them. The object owner does, however, automatically receive each of the privileges associated with the object (and can provide these privileges to other users, where supported, by using the WITH GRANT option of the GRANT statement). In addition, the object owner can alter, add a comment on, or drop the object. These authorizations are implicit for the object owner and cannot be revoked.

Related concepts:

- "Privileges, authority levels, and database authorities" on page 217
- "Schema privileges" on page 227
- "Table space privileges" on page 229
- "Table and view privileges" on page 229
- "Package privileges" on page 231
- "Index privileges" on page 232
- "Sequence privileges" on page 232
- "Routine privileges" on page 232

Details on privileges, authorities, and authorization

Each authority is discussed in this section followed by the different privileges.

System administration authority (SYSADM)

The SYSADM authority level 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, database partition groups, buffer pools, and event monitors.

SYSADM authority is assigned to the group specified by the *sysadm_group* configuration parameter. Membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSADM authority can perform the following functions:

- Migrate a database
- Change the database manager configuration file (including specifying the groups having SYSCTRL, SYSMAINT, or SYSMON authority)
- Grant DBADM authority.

Note: When a user with SYSADM authority creates a database, that user is automatically granted explicit DBADM authority on the database. If the database creator is removed from the SYSADM group and you want to prevent that user from accessing that database as a DBADM, you must explicitly revoke the user's DBADM authority.

Related concepts:

- "System control authority (SYSCTRL)" on page 222
- "System maintenance authority (SYSMAINT)" on page 222
- "Data encryption" on page 241
- "System monitor authority (SYSMON)" on page 224

System control authority (SYSCTRL)

SYSCTRL authority is the highest level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System control authority is designed for users administering a database manager instance containing sensitive data.

SYSCTRL authority is assigned to the group specified by the *sysctrl_group* configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSCTRL authority or higher can do the following:

- Update a database, node, or distributed connection services (DCS) directory
- Force users off the system
- Create or drop a database
- Drop, create, or alter a table space
- Restore to a new database.

In addition, a user with SYSCTRL authority can perform the functions of users with system maintenance authority (SYSMAINT) and system monitor authority (SYSMON).

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.

Related concepts:

- "System maintenance authority (SYSMAINT)" on page 222
- "Database administration authority (DBADM)" on page 223
- "System monitor authority (SYSMON)" on page 224

System maintenance authority (SYSMAINT)

SYSMAINT authority is the second level of system control authority. This authority provides the ability to perform maintenance and utility operations against the database manager instance and its databases. These operations can affect system resources, but they do not allow direct access to data in the databases. System maintenance authority is designed for users maintaining databases within a database manager instance that contains sensitive data.

SYSMAINT authority is assigned to the group specified by the *sysmaint_group* configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

Only a user with SYSMAINT or higher system authority can do the following:

- Update database configuration files
- 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, and can perform the functions of users with system monitor authority (SYSMON).

Related concepts:

- "Database administration authority (DBADM)" on page 223
- "System monitor authority (SYSMON)" on page 224

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_EXTERNAL_ROUTINE, CREATE_NOT_FENCED_ROUTINE, IMPLICIT_SCHEMA, QUIESCE_CONNECT, and LOAD database authorities 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.

Related concepts:

- "System administration authority (SYSADM)" on page 221
- "System control authority (SYSCTRL)" on page 222
- "System maintenance authority (SYSMAINT)" on page 222
- "LOAD authority" on page 225
- "Database authorities" on page 225
- "Implicit schema authority (IMPLICIT_SCHEMA) considerations" on page 227

System monitor authority (SYSMON)

SYSMON authority provides the ability to take database system monitor snapshots of a database manager instance or its databases. SYSMON authority is assigned to the group specified by the *sysmon_group* configuration parameter. If a group is specified, membership in that group is controlled outside the database manager through the security facility used on your platform.

SYSMON authority enables the user to run the following commands:

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

SYSMON authority enables the user to use the following APIs:

- db2GetSnapshot - Get Snapshot
- db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot() Output Buffer
- db2MonitorSwitches - Get/Update Monitor Switches
- db2ResetMonitor - Reset Monitor

SYSMON authority enables the user use the following SQL table functions:

- All snapshot table functions without previously running `SYSPROC.SNAPSHOT_FILEW`

  `SYSPROC.SNAPSHOT_FILEW` takes a snapshot and saves its content into a file. If any snapshot table functions are called with null input parameters, the file content is returned instead of a real-time system snapshot.

Users with the SYSADM, SYSCTRL, or SYSMAINT authority level also possess SYSMON authority.

Related reference:

- "sysmon_group - System monitor authority group name configuration parameter" in the *Administration Guide: Performance*

LOAD authority

Users having LOAD authority at the database level, as well as INSERT privilege on a table, can use the **LOAD** command to load data into a table.

Users having LOAD authority at the database level, as well as INSERT privilege on a table, can **LOAD RESTART** or **LOAD TERMINATE** if the previous load operation is a load to insert data.

Users having LOAD authority at the database level, as well as the INSERT and DELETE privileges on a table, can use the **LOAD REPLACE** command.

If the previous load operation was a load replace, the DELETE privilege must also have been granted to that user before the user can **LOAD RESTART** or **LOAD TERMINATE**.

If the exception tables are used as part of a load operation, the user must have INSERT privilege on the exception tables.

The user with this authority can perform **QUIESCE TABLESPACES FOR TABLE**, **RUNSTATS**, and **LIST TABLESPACES** commands.

Related concepts:

- "Privileges, authorities, and authorizations required to use Load" in the *Data Movement Utilities Guide and Reference*
- "Table and view privileges" on page 229

Related reference:

- "RUNSTATS Command" in the *Command Reference*
- "QUIESCE TABLESPACES FOR TABLE Command" in the *Command Reference*
- "LIST TABLESPACES Command" in the *Command Reference*
- "LOAD Command" in the *Command Reference*

Database authorities

Figure 4 on page 226 shows the database authorities.

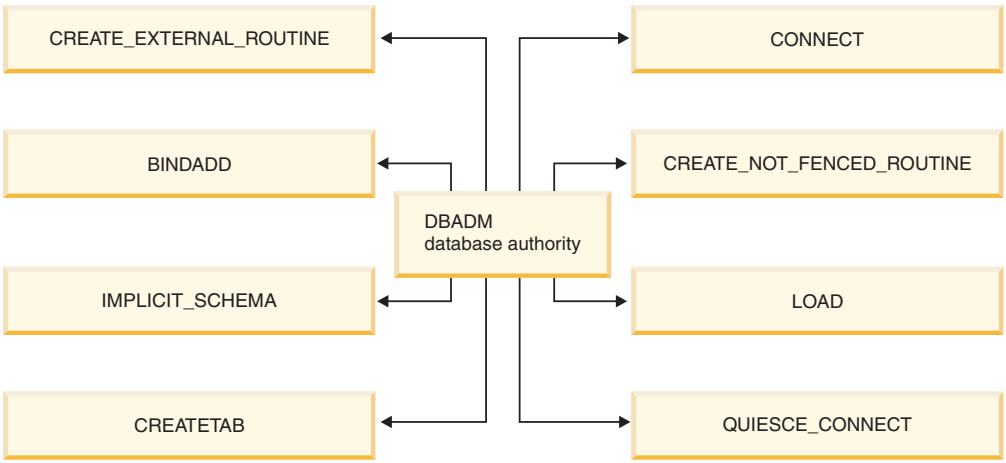


Figure 4. Database authorities

Database authorities involve actions on a database as a whole. Any user with DBADM authority possesses the complete set of database authorities, which are as follows:

- 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_EXTERNAL_ROUTINE allows a user to create a procedure for use by applications and other users of the database.
- CREATE_NOT_FENCED_ROUTINE 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.)

  Note: CREATE_EXTERNAL_ROUTINE is automatically granted to any user who is granted CREATE_NOT_FENCED_ROUTINE.
- 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.
- QUIESCE_CONNECT allows a user to access the database while it is quiesced.

Only users with SYSADM or DBADM authority can grant and revoke these database authorities to and from other users.

Note: When a database is created, the following database authorities are automatically granted to PUBLIC:

- CREATETAB database authority
- BINDADD database authority
- CONNECT database authority
- IMPLICIT_SCHEMA database authority
- USE privilege on USERSPACE1 table space
- SELECT privilege on the system catalog views.

To remove any database authority, a DBADM or SYSADM must explicitly revoke the database authority from PUBLIC.

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Implicit schema authority (IMPLICIT_SCHEMA) considerations

When a new database is created, PUBLIC is given IMPLICIT_SCHEMA database authority. With this authority, any user can create a schema by creating an object and specifying a schema name that does not already exist. SYSIBM becomes the owner of the implicitly created schema and PUBLIC is given the privilege to create objects in this schema.

If control of who can implicitly create schema objects is required for the database, IMPLICIT_SCHEMA database authority should be revoked from PUBLIC. Once this is done, there are only three (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.

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Schema privileges

Schema privileges are in the object privilege category. Object privileges are shown in Figure 5 on page 228.

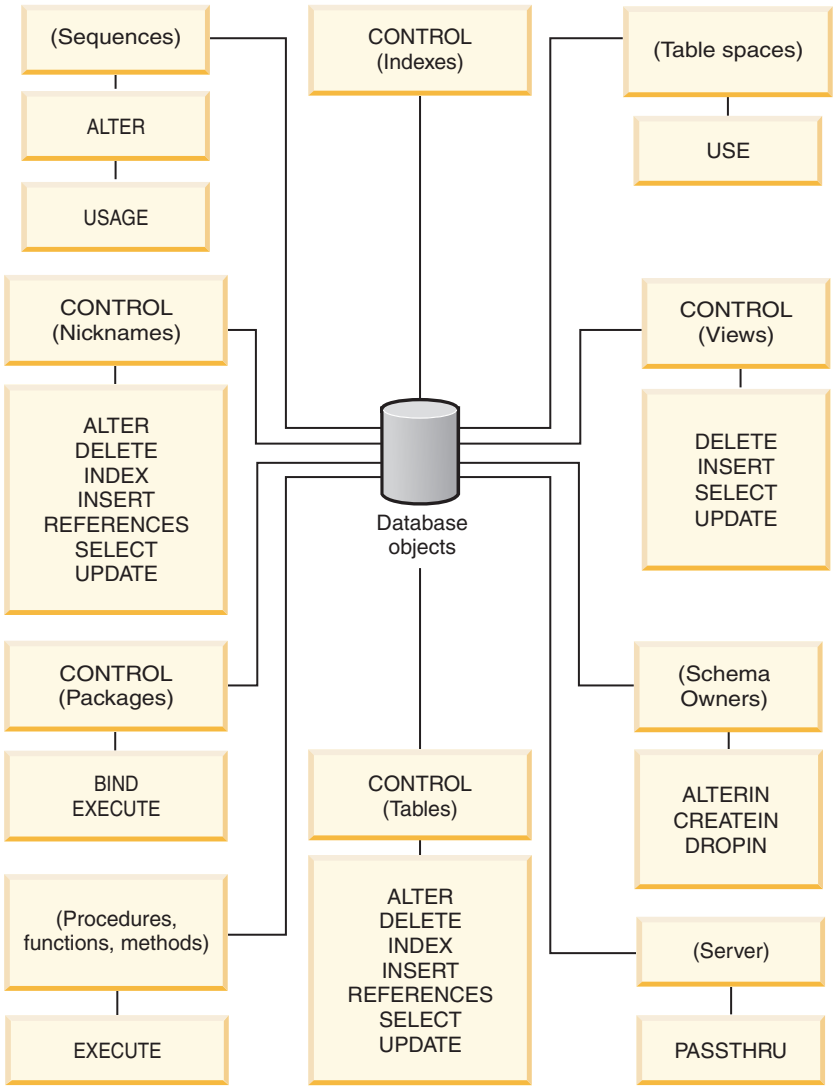


Figure 5. Object Privileges

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.

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Related reference:

- "ALTER SEQUENCE statement" in the *SQL Reference, Volume 2*

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.

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Related reference:

- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Table and view privileges

Table and view privileges involve actions on tables or views in a database. A user must have CONNECT authority 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, views, and nicknames referenced in the view definition, or they have SYSADM or DBADM authority.
- ALTER allows the user to modify on a table, for example, to add columns or a unique constraint to the table. A user with ALTER privilege can also COMMENT ON a table, or on columns of the table. For information about the possible modifications that can be performed on a table, see the ALTER TABLE and COMMENT statements.
- DELETE allows the user to delete rows from a table or view.
- INDEX allows the user to create an index on a table. Creators of indexes automatically have CONTROL privilege on the index.
- INSERT allows the user to insert 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.

Related concepts:

• "Index privileges" on page 232

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Related reference:

- "ALTER TABLE statement" in the *SQL Reference, Volume 2*
- "CREATE VIEW statement" in the *SQL Reference, Volume 2*
- "SELECT statement" in the *SQL Reference, Volume 2*

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 authority 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 also grant these privileges to other users by using the GRANT statement. (If a privilege is granted using WITH GRANT OPTION, a user who receives the BIND or EXECUTE privilege can, in turn, grant this privilege to other users.) To grant CONTROL privilege, the user must have SYSADM or DBADM authority.
- BIND privilege on a package allows the user to rebind or bind that package and to add new package versions of the same package name and creator.
- EXECUTE allows the user to execute or run a package.

Note: All package privileges apply to all VERSIONs that share the same package name and creator.

In addition to these package privileges, the BINDADD database privilege allows users to create new packages or rebind an existing package in the database.

Objects referenced by nicknames need to pass authentication checks at the data sources containing the objects. 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® Universal Database (DB2 UDB) 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.

Related concepts:

- "Database authorities" on page 225

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

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.

The nickname-level INDEX privilege allows a user to create an index specification on that nickname.

Related concepts:

- "Table and view privileges" on page 229

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Sequence privileges

The creator of a sequence automatically receives the USAGE and ALTER privileges on the sequence. The USAGE privilege is needed to use NEXT VALUE and PREVIOUS VALUE expressions for the sequence. To allow other users to use the NEXT VALUE and PREVIOUS VALUE expressions, sequence privileges must be granted to PUBLIC. This allows all users to use the expressions with the specified sequence.

ALTER privilege on the sequence allows the user to perform tasks such as restarting the sequence or changing the increment for future sequence values. The creator of the sequence can grant the ALTER privilege to other users, and if WITH GRANT OPTION is used, these users can, in turn, grant these privileges to other users.

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Related reference:

- "ALTER SEQUENCE statement" in the *SQL Reference, Volume 2*

Routine privileges

Execute privileges involve actions on all types of routines such as functions, procedures, and methods within a database. Once having EXECUTE privilege, a user can then invoke that routine, create a function that is sourced from that routine (applies to functions only), and reference the routine in any DDL statement such as CREATE VIEW or CREATE TRIGGER.

The user who defines the externally stored procedure, function, or method receives EXECUTE WITH GRANT privilege. If the EXECUTE privilege is granted to another user via WITH GRANT OPTION, that user can, in turn, grant the EXECUTE privilege to another user.

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

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.

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.

Note: A database group name must be 8 characters or less when used in a GRANT or REVOKE statement, or in the Control Center. Even though a database group name longer than 8 characters is accepted, the longer name results in an error message when users belonging to the group access database objects.

Related concepts:

- "Using the system catalog for security issues" on page 244

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Details on controlling access to database objects

The control of access to database objects is through the use of GRANT and REVOKE statements. Implicit access authorization is and indirect privileges are also discussed.

Granting privileges

Restrictions:

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.

Procedure:

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
```

Related concepts:

- "Controlling access to database objects" on page 233

Related tasks:

- "Revoking privileges" on page 234

Related reference:

- "GRANT (Database Authorities) statement" in the *SQL Reference, Volume 2*
- "GRANT (Index Privileges) statement" in the *SQL Reference, Volume 2*
- "GRANT (Package Privileges) statement" in the *SQL Reference, Volume 2*
- "GRANT (Schema Privileges) statement" in the *SQL Reference, Volume 2*
- "GRANT (Table, View, or Nickname Privileges) statement" in the *SQL Reference, Volume 2*
- "GRANT (Server Privileges) statement" in the *SQL Reference, Volume 2*
- "GRANT (Table Space Privileges) statement" in the *SQL Reference, Volume 2*
- "GRANT (Sequence Privileges) statement" in the *SQL Reference, Volume 2*
- "GRANT (Routine Privileges) statement" in the *SQL Reference, Volume 2*

Revoking privileges

The REVOKE statement allows authorized users to revoke privileges previously granted to other users.

Restrictions:

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 is not able to revoke a privilege that they granted through their use of the WITH GRANT OPTION. Also, there is no cascade on the revoke to those who have received privileges granted by the person being revoked.

If 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.

Procedure:

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).

Note: When CONTROL privilege is revoked from a user on a table or a view, the user continues to have the ability to grant privileges to others. When given CONTROL privilege, the user also receives all other privileges WITH GRANT OPTION. Once CONTROL is revoked, all of the other privileges remain WITH GRANT OPTION until they are explicitly revoked.

All packages that are dependent on revoked privileges are marked invalid, but can be validated if rebound by a user with appropriate authority. Packages can also be rebuilt if the privileges are subsequently granted again to the binder of the application; running the application will trigger a successful implicit rebind. If privileges are revoked from PUBLIC, all packages bound by users having only been able to bind based on PUBLIC privileges are invalidated. If DBADM authority is revoked from a user, all packages bound by that user are invalidated including those associated with database utilities. Attempting to use a package that

has been marked invalid causes the system to attempt to rebind the package. If this rebind attempt fails, an error occurs (SQLCODE -727). In this case, the packages must be explicitly rebound by a user with:

- Authority to rebind the packages
- Appropriate authority for the objects used within the packages

These packages should be rebound at the time the privileges are revoked.

If you define a trigger or SQL function based on one or more privileges and you lose one or more of these privileges, the trigger or SQL function cannot be used.

Related tasks:

- "Granting privileges" on page 233

Related reference:

- "REVOKE (Database Authorities) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Index Privileges) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Package Privileges) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Schema Privileges) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Table, View, or Nickname Privileges) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Server Privileges) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Table Space Privileges) statement" in the *SQL Reference, Volume 2*
- "REVOKE (Routine Privileges) statement" in the *SQL Reference, Volume 2*

Managing implicit authorizations by creating and dropping objects

Procedure:

The database manager implicitly grants certain privileges to a user creates a database object such as a table or a package. 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, nickname, 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, views, and nicknames 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.

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Establishing ownership of a package

Procedure:

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 Universal Database™ (DB2 UDB) database products support the OWNER option.

Related reference:

- "BIND Command" in the *Command Reference*
- "PRECOMPILE Command" in the *Command Reference*

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.

If the package includes dynamic SQL, the required privileges depend on the value that was specified for DYNAMICRULES when the package was precompiled or bound. For more information, see the topic that describes the effect of DYNAMICRULES on dynamic SQL.

Related concepts:

- "Indirect privileges through a package containing nicknames" on page 238
- "Effect of DYNAMICRULES bind option on dynamic SQL" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "BIND Command" in the *Command Reference*

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 and the nickname, 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® Universal Database (DB2 UDB) 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 UDB 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.

Related concepts:

• "Indirect privileges through a package" on page 237

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® Universal Database (DB2 UDB) 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, view, or nickname 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.

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, users of the view 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
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago

NAME	LOCATION
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

Related tasks:

- "Creating a view" on page 118
- "Granting privileges" on page 233

Monitoring access to data using the audit facility

The DB2® Universal Database (DB2 UDB) 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**.

Related concepts:

- "Introduction to the DB2 Universal Database (DB2 UDB) audit facility" on page 251

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.

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 VARCHAR FOR BIT DATA (with a limit of 32 631).

Only CHAR, VARCHAR, and FOR BIT DATA can be encrypted.

The DECRYPT_BIN and DECRYPT_CHAR functions decrypt data using password-based decryption.

DECRYPT_BIN always returns VARCHAR FOR BIT DATA while DECRYPT_CHAR always returns VARCHAR. Since the first argument may be CHAR FOR BIT DATA or VARCHAR FOR BIT DATA, there are cases where the result is not the same as the first argument.

The length of the result depends on the bytes to the next 8 byte boundary. The length of the result could be the length of the data argument plus 40 plus the number of bytes to the next 8 byte boundary when the optional hint parameter is specified. Or, the length of the result could be the length of the data argument plus 8 plus the number of bytes to the next 8 byte boundary when the optional hint parameter is not specified.

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.
- Encryption password special register. 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 ENCRYPTION PASSWORD special register is only stored in encrypted form.

  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.

Related reference:

- "SET ENCRYPTION PASSWORD statement" in the *SQL Reference, Volume 2*
- "DECRYPT_BIN and DECRYPT_CHAR scalar functions" in the *SQL Reference, Volume 1*
- "ENCRYPT scalar function" in the *SQL Reference, Volume 1*
- "GETHINT scalar function" in the *SQL Reference, Volume 1*

Tasks and required authorizations

Not all organizations divide job responsibilities in the same manner. Table 6 on page 243 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 6. 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. CREATE_EXTERNAL_ROUTINE may also be required.
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.
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).

Related concepts:

- "System administration authority (SYSADM)" on page 221
- "System control authority (SYSCTRL)" on page 222
- "System maintenance authority (SYSMAINT)" on page 222
- "Database administration authority (DBADM)" on page 223
- "LOAD authority" on page 225
- "Database authorities" on page 225

Related tasks:

- "Granting privileges" on page 233

- "Revoking privileges" on page 234

Using the system catalog for security issues

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.

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
SYSCAT.ROUTINEAUTH	Lists the routine (functions, methods, and stored procedures) privileges

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.

Related tasks:

- "Retrieving authorization names with granted privileges" on page 245
- "Retrieving all names with DBADM authority" on page 245
- "Retrieving names authorized to access a table" on page 245
- "Retrieving all privileges granted to users" on page 246
- "Securing the system catalog view" on page 247

Related reference:

- "SYSCAT.COLAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.DBAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.INDEXAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.PACKAGEAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.SCHEMAAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.TABAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.PASSTHRUAUTH catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.ROUTINEAUTH catalog view" in the *SQL Reference, Volume 1*

Details on using the system catalog for security issues

This section reviews some of the ways to determine who has what privileges within the database.

Retrieving authorization names with granted privileges

Procedure:

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.

Related concepts:

- "Using the system catalog for security issues" on page 244

Retrieving all names with DBADM authority

Procedure:

The following statement retrieves all authorization names that have been directly granted DBADM authority:

```
SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

Related concepts:

- "Database administration authority (DBADM)" on page 223
- "Using the system catalog for security issues" on page 244

Retrieving names authorized to access a table

Procedure:

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.

Related concepts:

- "Table and view privileges" on page 229
- "Using the system catalog for security issues" on page 244

Retrieving all privileges granted to users

Procedure:

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.

Related concepts:

- "Privileges, authority levels, and database authorities" on page 217
- "Database authorities" on page 225
- "Using the system catalog for security issues" on page 244

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Securing the system catalog view

Procedure:

During database creation, SELECT privilege on the system catalog views is granted to PUBLIC. In most cases, this does not present any security problems. For very sensitive data, 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. 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
```

Related concepts:

- "Catalog statistics" in the *Administration Guide: Performance*
- "Database authorities" on page 225
- "Using the system catalog for security issues" on page 244

Related tasks:

- "Granting privileges" on page 233
- "Revoking privileges" on page 234

Introduction to firewall support

A *firewall* is a set of related programs, located at a network gateway server, that are used to prevent unauthorized access to a system or network.

There are four types of firewalls:

1. Network level, packet-filter, or screening router firewalls
2. Classical application level proxy firewalls
3. Circuit level or transparent proxy firewalls
4. Stateful multi-layer inspection (SMLI) firewalls

There are existing firewall products that incorporate one of the firewall types listed above. There are many other firewall products that incorporate some combination of the above types.

Related concepts:

- "Screening router firewalls" on page 248
- "Application proxy firewalls" on page 249
- "Circuit level firewalls" on page 249
- "Stateful multi-layer inspection (SMLI) firewalls" on page 249

Screening router firewalls

This type of firewall is also known as a network level or packet-filter firewall. Such a firewall works by screening incoming packets by protocol attributes. The protocol attributes screened may include source or destination address, type of protocol, source or destination port, or some other protocol-specific attributes.

For all firewall solutions (except SOCKS), you need to ensure that all the ports used by DB2® Universal Database (DB2 UDB) are open for incoming and outgoing packets. DB2 UDB uses port 523 for the DB2 Administration Server (DAS), which is used by the DB2 UDB tools. Determine the ports used by all your server

instances by using the services file to map the service name in the server database manager configuration file to its port number.

Related concepts:

- "Introduction to firewall support" on page 248

Application proxy firewalls

A proxy or proxy server is a technique that acts as an intermediary between a Web client and a Web server. A proxy firewall acts as a gateway for requests arriving from clients. When client requests are received at the firewall, the final server destination address is determined by the proxy software. The application proxy translates the address, performs additional access control checking and logging as necessary, and connects to the server on behalf of the client.

The DB2® Connect product on a firewall machine can act as a proxy to the destination server. Also, a DB2 Universal Database™ (DB2 UDB) server on the firewall, acting as a hop server to the final destination server, acts like an application proxy.

Related concepts:

- "Introduction to firewall support" on page 248

Circuit level firewalls

This type of firewall is also known as a transparent proxy firewall. A transparent proxy firewall does not modify the request or response beyond what is required for proxy authentication and identification. An example of a transparent proxy firewall is SOCKS.

DB2® Universal Database (DB2 UDB) supports SOCKS Version 4.

Related concepts:

- "Introduction to firewall support" on page 248

Stateful multi-layer inspection (SMLI) firewalls

This type of firewall is a sophisticated form of packet-filtering that examines all seven layers of the Open System Interconnection (OSI) model. Each packet is examined and compared against known states of friendly packets. While screening router firewalls only examine the packet header, SMLI firewalls examine the entire packet including the data.

Related concepts:

- "Introduction to firewall support" on page 248

Chapter 8. Auditing DB2 Universal Database™ (DB2 UDB) activities

Introduction to the DB2 Universal Database (DB2 UDB) audit facility

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® Universal Database (DB2 UDB) 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 UDB 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.

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 UDB 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 UDB instance.
- Stop recording auditable events within the DB2 UDB 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.

Note: Ensure that the audit facility has been turned on by issuing the db2audit start command before using the audit utilities.

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 UDB 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.

Related concepts:

- "Audit facility behavior" on page 253

- "Audit facility record layouts (introduction)" on page 266
- "Audit facility tips and techniques" on page 282

Related tasks:

- "Controlling DB2 UDB audit facility activities" on page 283

Related reference:

- "Audit facility usage" on page 254
- "Audit facility messages" on page 266

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® Universal Database (DB2 UDB) that can operate even if the DB2 UDB 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 UDB 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 UDB 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 UDB. 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.

Depending on the API or SQL statement and the audit settings for the DB2 UDB 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.

Related concepts:

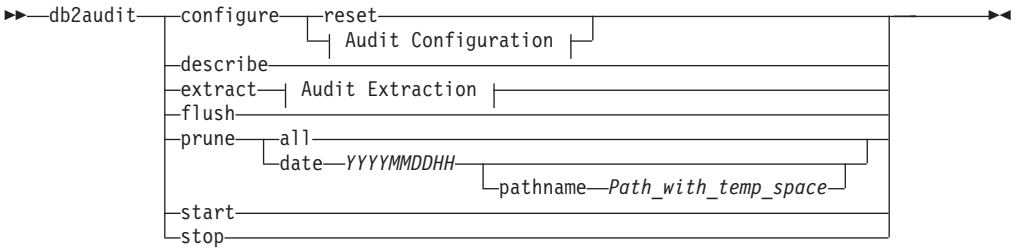
- "Introduction to the DB2 Universal Database (DB2 UDB) audit facility" on page 251

Related reference:

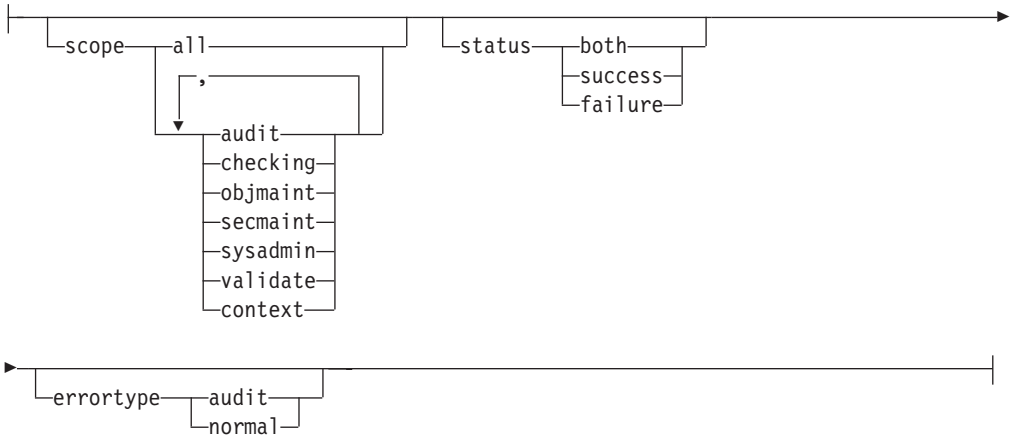
- "audit_buf_sz - Audit buffer size configuration parameter" in the *Administration Guide: Performance*
- "Audit facility usage" on page 254

Audit facility usage

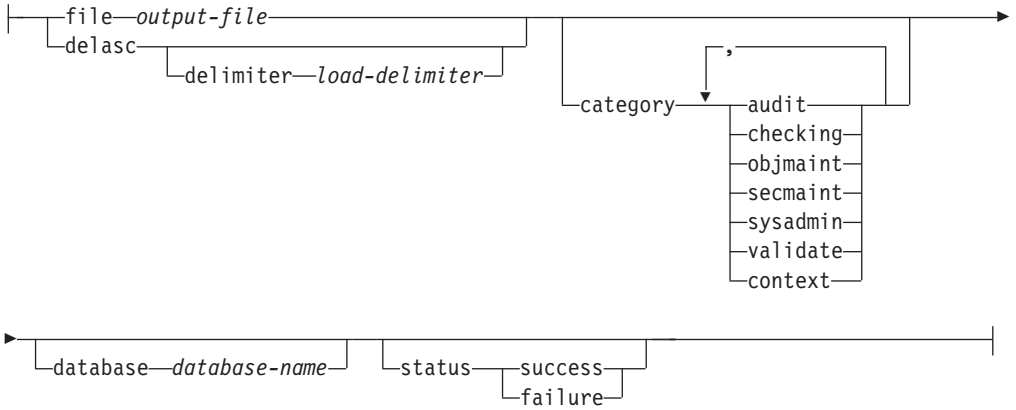
A review of each part of the following syntax diagrams will assist you in the understanding of how the audit facility can be used.



Audit Configuration:



Audit Extraction:



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 Universal Database™ (DB2 UDB) 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 UDB 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, all of the audit records are extracted and placed in a flat report file. 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*). If no file name is specified, records are written to the `db2audit.out` file in the `security` subdirectory of `sqllib`. If no directory is specified, *output_file* is written to the current working directory.
- DELASC. The extracted audit records are placed in a delimited ASCII format suitable for loading into DB2 UDB 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`

These files are always written to the `security` subdirectory of `sqllib`.

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).

- 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 UDB 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 UDB 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.

Related concepts:

- "Introduction to the DB2 Universal Database (DB2 UDB) audit facility" on page 251
- "Audit facility tips and techniques" on page 282

Related reference:

- "db2audit - Audit Facility Administrator Tool Command" in the *Command Reference*

Working with DB2 audit data in DB2 tables

The following topics describe how to create DB2 audit data, how to create tables to hold this data, how to populate the tables with the DB2 audit data, and how to select the DB2 audit data from the tables.

Working with DB2 audit data in DB2 tables

When you use the DB2 audit facility to maintain an audit trail of database activities, by default the audit facility places the audit records in a log file. If you want, you can write the audit records from the log file to a text file, or you can write the audit records from the log file to delimited ASCII files, then load the contents of the ASCII files into DB2 tables. When the audit data is in DB2 tables, you can select the data from the tables to answer questions that you may have about activity on your DB2 instance.

Procedure:

To work with audit data in DB2 tables:

1. Create tables to hold the DB2 audit data.
2. Create the DB2 audit data files.
3. Use the load utility to populate the tables with the data.
4. Select the table data.

Related concepts:

- "Audit facility behavior" on page 253
- "Audit facility tips and techniques" on page 282

Related tasks:

- "Creating DB2 audit data files" on page 261
- "Creating tables to hold the DB2 audit data" on page 258
- "Loading DB2 audit data into tables" on page 263
- "Selecting DB2 audit data from tables" on page 265

Related reference:

- "Audit facility usage" on page 254

Creating tables to hold the DB2 audit data

Before you can work with audit data in tables, you need to create the tables to hold the data. You should consider creating these tables in a separate schema to isolate the data in the tables from unauthorized users.

Prerequisites:

- See the CREATE SCHEMA statement for the authorities and privileges that you require to create a schema.
- See the CREATE TABLE statement for the authorities and privileges that you require to create a table.

- Decide which table space you want to use to hold the tables. (This topic does not describe how to create table spaces.)

Procedure:

The examples that follow show how to the create tables that will hold all of the records from all of the ASCII files. If you want, you can create a separate schema to contain these tables.

If you do not want to use all of the data that is contained in the files, you can omit columns from the table definitions, or bypass creating tables, as required. If you omit columns from the table definitions, you must modify the commands that you use to load data into these tables.

1. Issue the **db2** command to open a DB2 command window.
2. Optional. Create a schema to hold the tables. Issue the following command. For this example, the schema is called `AUDIT`

   ```
   CREATE SCHEMA AUDIT
   ```

3. Optional. If you created the `AUDIT` schema, switch to the schema before creating any tables. Issue the following command:

   ```
   SET CURRENT SCHEMA = 'AUDIT'
   ```

4. To create the table that will contain records from the `audit.del` file, issue the following SQL statement:

   ```
   CREATE TABLE AUDIT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128))
   ```

5. To create the table that will contain records from the `checking.del` file, issue the following SQL statement:

   ```
   CREATE TABLE CHECKING (TIMESTAMP CHAR(26),
                          CATEGORY CHAR(8),
                          EVENT VARCHAR(32),
                          CORRELATOR INTEGER,
                          STATUS INTEGER,
                          DATABASE CHAR(8),
                          USERID VARCHAR(1024),
                          AUTHID VARCHAR(128),
                          NODENUM SMALLINT,
                          COORDNUM SMALLINT,
                          APPID VARCHAR(255),
                          APPNAME VARCHAR(1024),
                          PKGSCHEMA VARCHAR(128),
                          PKGNAME VARCHAR(128),
                          PKGSECNUM SMALLINT,
                          OBJSCHEMA VARCHAR(128),
                          OBJNAME VARCHAR(128),
                          OBJTYPE VARCHAR(32),
                          ACCESSAPP CHAR(18),
                          ACCESSATT CHAR(18),
                          PKGVER VARCHAR(64))
   ```

6. To create the table that will contain records from the `objmaint.del` file, issue the following SQL statement:

   ```
   CREATE TABLE OBJMAINT (TIMESTAMP CHAR(26),
                          CATEGORY CHAR(8),
                          EVENT VARCHAR(32),
                          CORRELATOR INTEGER,
                          STATUS INTEGER,
   ```

```
                                DATABASE CHAR(8),
                                USERID VARCHAR(1024),
                                AUTHID VARCHAR(128),
                                NODENUM SMALLINT,
                                COORDNUM SMALLINT,
                                APPID VARCHAR(255),
                                APPNAME VARCHAR(1024),
                                PKGSCHEMA VARCHAR(128),
                                PKGNAME VARCHAR(128),
                                PKGSECNUM SMALLINT,
                                OBJSCHEMA VARCHAR(128),
                                OBJNAME VARCHAR(128),
                                OBJTYPE VARCHAR(32),
                                PACKVER VARCHAR(64))
```

7. To create the table that will contain records from the `secmaint.del` file, issue the following SQL statement:

```
CREATE TABLE SECMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       GRANTOR VARCHAR(128),
                       GRANTEE VARCHAR(128),
                       GRANTEETYPE VARCHAR(32),
                       PRIVAUTH CHAR(18),
                       PKGVER VARCHAR(64))
```

8. To create the table that will contain records from the `sysadmin.del` file, issue the following SQL statement:

```
CREATE TABLE SYSADMIN (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64))
```

9. To create the table that will contain records from the `validate.del` file, issue the following SQL statement:

```
CREATE TABLE VALIDATE (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
```

```
                                DATABASE CHAR(8),
                                USERID VARCHAR(1024),
                                AUTHID VARCHAR(128),
                                EXECID VARCHAR(1024),
                                NODENUM SMALLINT,
                                COORDNUM SMALLINT,
                                APPID VARCHAR(255),
                                APPNAME VARCHAR(1024),
                                AUTHTYPE VARCHAR(32),
                                PKGSCHEMA VARCHAR(128),
                                PKGNAME VARCHAR(128),
                                PKGSECNUM SMALLINT,
                                PKGVER VARCHAR(64)
                                PLUGINNAME VARCHAR(32))
```

10. To create the table that will contain records from the `context.del` file, issue the following SQL statement:

```
CREATE TABLE CONTEXT (TIMESTAMP CHAR(26),
                      CATEGORY CHAR(8),
                      EVENT VARCHAR(32),
                      CORRELATOR INTEGER,
                      DATABASE CHAR(8),
                      USERID VARCHAR(1024),
                      AUTHID VARCHAR(128),
                      NODENUM SMALLINT,
                      COORDNUM SMALLINT,
                      APPID VARCHAR(255),
                      APPNAME VARCHAR(1024),
                      PKGSCHEMA VARCHAR(128),
                      PKGNAME VARCHAR(128),
                      PKGSECNUM SMALLING,
                      STMTTEXT CLOB(2M),
                      PKGVER VARCHAR(64))
```

11. After creating the tables, issue the COMMIT statement to ensure that the table definitions are written to disk.
12. When you have created the tables, you are ready to extract the audit records from the db2audit.log file to delimited ASCII files.

Related tasks:

- "Setting a schema" on page 82

Related reference:

- "CREATE SCHEMA statement" in the *SQL Reference, Volume 2*
- "CREATE TABLE statement" in the *SQL Reference, Volume 2*

Creating DB2 audit data files

By default, the DB2 audit facility writes audit data to the `db2audit.log` file. The records in this file cannot be loaded into tables. You must extract the audit records to delimited ASCII files, which can you use to populate tables.

Prerequisites:

You require SYSADM authority to use the **db2audit** command.

Procedure:

To write the audit facility records to delimited ASCII files:

1. Review the topic on audit facility usage to determine the type of DB2 activities that you want to audit. When you are satisfied with the configuration that you have set up for the audit facility, issue the following command to begin auditing:

   ```
   db2audit start
   ```

2. Issue the following command to ensure that all audit records are flushed from memory to the `db2audit.log` file:

   ```
   db2audit flush
   ```

3. Issue the following command to move the audit records from the `db2audit.log` to delimited ASCII files:

   ```
   db2audit extract delasc
   ```

   The following files are created in the `security` subdirectory of `sqllib`. If you are not auditing a particular type of event, the file for that event is created, but the file is empty.
   - `audit.del`
   - `checking.del`
   - `objmaint.del`
   - `secmaint.del`
   - `sysadmin.del`
   - `validate.del`
   - `context.del`

4. Issue the following command to delete the audit records from the `db2audit.log` file that you just extracted:

   ```
   db2audit prune date YYYYMMDDHH
   ```

   Where *YYYYMMDDHH* is the current year, month, day, and hour. Write down the value that you use because you will require this information in the next step when you populate the tables with the audit data.

   The audit facility will continue to write new audit records to the `db2audit.log` file, and these records will have a timestamp that is later than *YYYYMMDDHH*. Pruning records from the `db2audit.log` file that you have already extracted prevents you from extracting the same records a second time. All audit records that are written after *YYYYMMDDHH* will be written to the `.del` files the next time you extract the audit data.

5. After you create the audit data files, the next step is to use the load utility to populate the tables with the audit data.

Related reference:

- "db2audit - Audit Facility Administrator Tool Command" in the *Command Reference*
- "Audit facility usage" on page 254
- "Audit record layout for AUDIT events" on page 267
- "Audit record layout for CHECKING events" on page 268
- "Audit record layout for OBJMAINT events" on page 273
- "Audit record layout for SECMAINT events" on page 274
- "Audit record layout for SYSADMIN events" on page 278
- "Audit record layout for VALIDATE events" on page 279
- "Audit record layout for CONTEXT events" on page 281

Loading DB2 audit data into tables

When you have created the tables to hold the audit data, you then load the data in the ASCII files into the tables.

Prerequisites:

See the topic on the privileges, authorities, and authorizations required to use the load utility for more information.

Procedure:

Use the load utility to load the data into the tables. Issue a separate load command for each table. If you omitted one or more columns from the table definitions, you must modify the version of the LOAD command that you use to successfully load the data. Also, if you specified a delimiter character other than the default (`0xff`) when you extracted the audit data, you must also modify the version of the LOAD command that you use (see the topic ″ File type modifiers for load″ for more information).

1. Issue the **db2** command to open a DB2 command window.
2. To load the AUDIT table, issue the following command:

   ```
   LOAD FROM audit.del OF del MODIFIED BY CHARDEL0xff INSERT INTO schema.AUDIT
   ```

   Note: When specifying the file name, usen the fully qualified path name. For example, if you have DB2 UDB installed on the C: drive of a Windows-based computer, you would specify `C:\Program Files\IBM\SQLLIB\`*instance*`\security\audit.del` as the fully qualified file name for the `audit.del` file.

   After loading the AUDIT table, issue the following DELETE statement to ensure that you do not load duplicate rows into the table the next time you load it. When you extracted the audit records from the `db2audit.log` file, all records in the file were written to the `.del` files. Likely, the `.del` files contained records that were written after the hour to which the audit log was subsequently pruned (because the **db2audit prune** command only prunes records to a specified hour). The next time you extract the audit records, the new `.del` files will contain records that were previously extracted, but not deleted by the **db2audit prune** command (because they were written after the hour specified for the prune operation). Deleting rows from the table to the same hour to which the `db2audit.log` file was pruned ensures that the table does not contain duplicate rows, and that no audit records are lost.

   ```
   DELETE FROM schema.AUDIT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   Where *YYYYMMDDHH* is the value that you specified when you pruned the `db2audit.log` file. Because the DB2 audit facility continues to write audit records to the `db2audit.log` file after it is pruned, you must specify `0000` for the minutes and seconds to ensure that audit records that were written after the `db2audit.log` file was pruned are not deleted from the table.
3. To load the CHECKING table, issue the following command:

   ```
   LOAD FROM checking.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
     schema.CHECKING
   ```

   After loading the CHECKING table, issue the following SQL statement to ensure that you do not load duplicate rows into the table the next time you load it:

   ```
   DELETE FROM schema.CHECKING WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

Where *YYYYMMDDHH* is the value that you specified when you pruned the log file.

4. To load the OBJMAINT table, issue the following command:

   ```
   LOAD FROM objmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
      schema.OBJMAINT
   ```

   After loading the OBJMAINT table, issue the following SQL statement to ensure that you do not load duplicate rows into the table the next time you load it:

   ```
   DELETE FROM schema.OBJMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   Where *YYYYMMDDHH* is the value that you specified when you pruned the log file.

5. To load the SECMAINT table, issue the following command:

   ```
   LOAD FROM secmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
      schema.SECMAINT
   ```

   After loading the SECMAINT table, issue the following SQL statement to ensure that you do not load duplicate rows into the table the next time you load it:

   ```
   DELETE FROM schema.SECMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   Where *YYYYMMDDHH* is the value that you specified when you pruned the log file.

6. To load the SYSADMIN table, issue the following command:

   ```
   LOAD FROM sysadmin.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
      schema.SYSADMIN
   ```

   After loading the SYSADMIN table, issue the following SQL statement to ensure that you do not load duplicate rows into the table the next time you load it:

   ```
   DELETE FROM schema.SYSADMIN WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   Where *YYYYMMDDHH* is the value that you specified when you pruned the log file.

7. To load the VALIDATE table, issue the following command:

   ```
   LOAD FROM validate.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
      schema.VALIDATE
   ```

   After loading the VALIDATE table, issue the following SQL statement to ensure that you do not load duplicate rows into the table the next time you load it:

   ```
   DELETE FROM schema.VALIDATE WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   Where *YYYYMMDDHH* is the value that you specified when you pruned the log file.

8. To load the CONTEXT table, issue the following command:

   ```
   LOAD FROM context.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
      schema.CONTEXT
   ```

   After loading the CONTEXT table, issue the following SQL statement to ensure that you do not load duplicate rows into the table the next time you load it:

   ```
   DELETE FROM schema.CONTEXT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   Where *YYYYMMDDHH* is the value that you specified when you pruned the log file.

9. After you finish loading the data into the tables, delete the `.del` files from the `security` subdirectory of the `sqllib` directory.
10. When you have loaded the audit data into the tables, you are ready to select data from these tables.

If you have already populated the tables a first time, and want to do so again, use the INSERT option to have the new table data added to the existing table data. If you want to have the records from the previous **db2audit extract** operation removed from the tables, load the tables again using the REPLACE option. In either situation, remember both to flush the audit records to the `db2audit.log` file before extracting the records to the `.del` files, and to prune the `db2audit.log` file after extracting the records so that you do not load the same records into the tables more than once.

Related tasks:

- "Working with DB2 audit data in DB2 tables" on page 258

Selecting DB2 audit data from tables

When the audit data is successfully loaded into the tables, you can select data from these tables for further analysis.

Prerequisites:

See the topic on the SELECT statement for information about the authorities and privileges required to select data from a table.

Procedure:

To select all the rows in a table:

1. Issue the **db2** command to open a DB2 command window.
2. Issue an SQL statement of the following form for each table from which you want to select audit data:

```
SELECT * FROM schema.table
```

For example, to select all the data from the `CHECKING` table in the `AUDIT` schema, use the following statement:

```
SELECT * FROM AUDIT.CHECKING
```

The select that you perform should reflect the type of analysis that you want to do on the data. For example, you can select records according to an authorization ID (authid) to determine the type of activities that this authorization ID has been performing:

```
SELECT * FROM AUDIT.CHECKING WHERE AUTHID = authorization ID
```

Where *authorization ID* is the user ID for which you want to analyze the data.

For a description of the values that can be included in audit data, see the corresponding audit record layout topic for the table, and the list of possible returned values for the table.

Related reference:

- "Subselect" in the *SQL Reference, Volume 1*
- "SELECT statement" in the *SQL Reference, Volume 2*

Audit facility messages

SQL1322N An error occurred when writing to the audit log file.

Explanation: The DB2 Universal Database™ (DB2 UDB) 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

Related concepts:

- "Introduction to the DB2 Universal Database (DB2 UDB) audit facility" on page 251

Audit facility record layouts (introduction)

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.

3. A new field called "Package Version" has been added to the record layout for the CHECKING, OBJMAINT, SECMAINT, SYSADMIN, VALIDATE, and CONTEXT events.

Related reference:

- "Audit record layout for AUDIT events" on page 267
- "Audit record layout for CHECKING events" on page 268
- "Audit record layout for OBJMAINT events" on page 273
- "Audit record layout for SECMAINT events" on page 274
- "Audit record layout for SYSADMIN events" on page 278
- "Audit record layout for VALIDATE events" on page 279
- "Audit record layout for CONTEXT events" on page 281

Details on audit facility record layouts

The various audit facility record layouts are shown in this section.

Audit record layout for AUDIT events

Table 7. 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.

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Audit record layout for CHECKING events

Table 8. 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;
  package schema=NULLID;package name=SYSSH200;
  package section=0;object schema=GSTAGER;object name=NONE;object type=REOPT_VALUES;
  access approval reason=DBADM;access attempted=STORE;
```

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 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: those shown in the topic titled “Audit record object types”.
Access Approval Reason	CHAR(18)	Indicates the reason why access was approved for this audit event. Possible values include: those shown in the topic titled “List of possible CHECKING access approval reasons”.
Access Attempted	CHAR(18)	Indicates the type of access that was attempted. Possible values include: those shown in the topic titled “List of possible CHECKING access attempted types”.

Table 8. 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;
  package schema=NULLID;package name=SYSSH200;
  package section=0;object schema=GSTAGER;object name=NONE;object type=REOPT_VALUES;
  access approval reason=DBADM;access attempted=STORE;
```

NAME	FORMAT	DESCRIPTION
Package Version	VARCHAR (64)	Version of the package in use at the time that the audit event occurred.

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Related reference:

- "List of possible CHECKING access approval reasons" on page 270
- "List of possible CHECKING access attempted types" on page 271
- "Audit record object types" on page 269

Audit record object types

Table 9. Audit Record Object Types Based on Audit Events

Object type	CHECKING events	OBJMAINT events	SECMAINT events
NONE	X	X	X
TABLE	X	X	X
VIEW	X	X	X
ALIAS	X	X	
FUNCTION	X	X	X
INDEX	X	X	X
INDEX EXTENSION		X	
PACKAGE	X	X	X
PACKAGE CACHE	X		
DATA_TYPE		X	
NODEGROUP	X	X	
SCHEMA	X	X	X
STORED_PROCEDURE	X	X	X
METHOD_BODY	X	X	X
BUFFERPOOL	X	X	
SEQUENCE	X	X	
TABLESPACE	X	X	X
EVENT_MONITOR	X	X	
TRIGGER		X	
DATABASE	X		X
INSTANCE	X		
FOREIGN_KEY		X	
PRIMARY_KEY		X	

Table 9. Audit Record Object Types Based on Audit Events (continued)

Object type	CHECKING events	OBJMAINT events	SECMAINT events
UNIQUE_CONSTRAINT		X	
CHECK_CONSTRAINT		X	
WRAPPER	X	X	
SERVER	X	X	X
NICKNAME	X	X	X
USER MAPPING	X	X	
SERVER OPTION	X	X	
TYPE&TRANSFORM	X	X	
TYPE MAPPING	X	X	
FUNCTION MAPPING	X	X	
SUMMARY TABLES	X	X	X
JAR_FILE		X	
ALL	X		
REOPT_VALUES	X		

Related reference:

- "Audit record layout for CHECKING events" on page 268
- "Audit record layout for OBJMAINT events" on page 273
- "Audit record layout for SECMAINT events" on page 274

List of possible CHECKING access approval reasons

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.

0x0000000000000800 SYSQUIESCE
Access is approved; if the instance or database is in quiesce mode, the application/user may connect or attach.

0x0000000000001000 SYSMON
Access is approved; the application/user has SYSMON authority.

Related reference:

- "Audit record layout for CHECKING events" on page 268
- "List of possible CHECKING access attempted types" on page 271

List of possible CHECKING access attempted types

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 or to invoke a routine, create a function sourced from the routine (applies to functions only), or reference a routine in any DDL statement.

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.

0x0000000010000000 SET SESSION_USER
Attempt to execute the SET SESSION_USER statement.

0x0000000020000000 FLUSH
Attempt to execute the FLUSH statement.

0x0000000040000000 STORE
Attempt to view the values of a re-optimized statement in the EXPLAIN_PREDICATE table.

Related reference:

- "Audit record layout for CHECKING events" on page 268
- "List of possible CHECKING access approval reasons" on page 270

Audit record layout for OBJMAINT events

Table 10. 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.
Object Type	VARCHAR (32)	Type of object for which the audit event was generated. Possible values include: those shown in the topic titled “Audit record object types”.
Package Version	VARCHAR (64)	Version of the package in use at the time the audit event occurred.

Related concepts:

- “Introduction to the DB2 Universal Database (DB2 UDB) audit facility” on page 251

Related reference:

- "Audit record object types" on page 269

Audit record layout for SECMAINT events

Table 11. 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, SET_SESSION_USER, 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.
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: those shown in the topic titled "Audit record object types".
Grantor	VARCHAR (128)	Grantor ID.
Grantee	VARCHAR (128)	Grantee ID for which a privilege or authority was granted or revoked.

Table 11. 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
Grantee Type	VARCHAR (32)	Type of the grantee that was granted to or revoked from. Possible values include: USER, GROUP, or BOTH.
Privilege or Authority	CHAR(18)	Indicates the type of privilege or authority granted or revoked. Possible values include: those shown in the topic titled "List of possible SECMAINT privileges or authorities".
Package Version	VARCHAR (64)	Version of the package in use at the time the audit event occurred.

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Related reference:

- "List of possible SECMAINT privileges or authorities" on page 275
- "Audit record object types" on page 269

List of possible SECMAINT privileges or authorities

The following is the list of possible SECMAINT privileges or authorities:

0x0000000000000001 Control Table
Control privilege granted or revoked on a table or view.

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 or view.

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 or view.

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 or view.

0x0000000000001000 Table UPDATE with GRANT
Privilege granted or revoked on an update on a table or view 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.

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.

0x0000008000000000 Package BIND
BIND privilege granted or revoked on a package.

0x0000010000000000 Package BIND with GRANT
BIND privilege granted or revoked on a package with granting of privileges allowed.

0x0000020000000000 EXECUTE
EXECUTE privilege granted or revoked on a package or a routine.

0x0000040000000000 EXECUTE with GRANT
EXECUTE privilege granted or revoked on a package or a routine with granting of privileges allowed.

0x0000080000000000 EXECUTE IN SCHEMA
EXECUTE privilege granted or revoked for all routines in a schema.

0x0000100000000000 EXECUTE IN SCHEMA with GRANT
EXECUTE privilege granted or revoked for all routines in a schema with granting of privileges allowed.

0x0000020000000000 EXECUTE IN TYPE
EXECUTE privilege granted or revoked for all routines in a type.

0x0000400000000000 EXECUTE IN TYPE with GRANT
EXECUTE privilege granted or revoked for all routines in a type with granting of privileges allowed.

0x0000080000000000 CREATE EXTERNAL ROUTINE
CREATE EXTERNAL ROUTINE privilege granted or revoked.

0x0001000000000000 QUIESCE_CONNECT
QUIESCE_CONNECT privilege granted or revoked.

Related reference:

- "Audit record layout for SECMAINT events" on page 274

Audit record layout for SYSADMIN events

Table 12. 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
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.
Package Version	VARCHAR (64)	Version of the package in use at the time the audit event occurred.

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Related reference:

- "List of possible SYSADMIN audit events" on page 278

List of possible SYSADMIN audit events

The following is the list of possible SYSADMIN audit events:

Table 13. 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

Related reference:

- "Audit record layout for SYSADMIN events" on page 278

Audit record layout for VALIDATE events

Table 14. 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.

Table 14. Audit Record Layout for VALIDATE Events (continued)

`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
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.
Package Version	VARCHAR (64)	Version of the package in use at the time the audit event occurred.
Plug-in Name	VARCHAR(32)	The name of the plug-in in use at the time the audit event occurred.

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Audit record layout for CONTEXT events

Table 15. 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 (2M)	Text of the SQL statement, if applicable. Null if no SQL statement text is available.
Package Version	VARCHAR (64)	Version of the package in use at the time the audit event occurred.

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Related reference:

- "List of possible CONTEXT audit events" on page 281

List of possible CONTEXT audit events

The following is the list of possible CONTEXT audit events:

Table 16. 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

Related reference:

- "Audit record layout for CONTEXT events" on page 281

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® Universal Database (DB2 UDB) 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".

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Related reference:

- "Audit facility usage" on page 254

Controlling DB2 UDB audit facility activities

Procedure:

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)
```

Related concepts:

- "Audit facility record layouts (introduction)" on page 266

Related reference:

- "Audit facility usage" on page 254

Part 3. Appendixes

Appendix A. Conforming to the naming rules

General naming rules

Rules exist 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 Windows® platforms, names can be in upper, lower, and mixed-case.

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.
- ! % ( ) { } . – ^ ~ _ (underscore) @, #, $, and space.
- \ (backslash).

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.

There are other special characters that might work separately depending on your operating system and where you are working with DB2® Universal Database (DB2 UDB). 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.

You also need to consider object naming rules, workstation naming rules, naming rules in an NLS environment, and naming rules in a Unicode environment.

Related concepts:

- "DB2 UDB object naming rules" on page 291
- "Workstation naming rules" on page 296
- "User, user ID and group naming rules" on page 294
- "Federated database object naming rules" on page 294

DB2 UDB object naming rules

All objects follow the General Naming Rules. In addition, some objects have additional restrictions shown in the accompanying tables.

Table 17. 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® Universal Database (DB2 UDB), this location is a directory path, while on Windows® implementations, it is a logical disk. • 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®, Windows 2000, Windows XP and Windows Server 2003 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 18. Database object naming rules

Objects	Guidelines
• Aliases • Buffer pools • Columns • Event monitors • Indexes • Methods • Nodegroups • Packages • Package versions • 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 can contain up to 30 bytes • Package names, which can contain up to 8 bytes • Schema names, which can contain up to 30 bytes • Package versions, which can contain up to 64 bytes • Object names can also include: – valid accented characters (such as ö) – multibyte characters, except multibyte spaces (for multibyte environments) • Package names and package versions can also include periods (.), hyphens (-), and colons (:).

Table 19. 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)

Delimited identifiers and 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.

Additional schema names information:

- 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.

Related concepts:

- "General naming rules" on page 291

Delimited identifiers and 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.

Related concepts:

- "General naming rules" on page 291

User, user ID and group naming rules

Table 20. User, user ID and group naming rules

Objects	Guidelines
• Group names • User names • User IDs	• Group names can contain up to 30 characters. • User IDs on Linux and UNIX®-based systems can contain up to 8 characters. • User names on Windows® can contain up to 30 characters. Windows NT®, Windows 2000, Windows XP and Windows Server 2003 currently have a practical limit of 20 characters. • When not using Client authentication, non-Windows 32-bit clients connecting to Windows NT, Windows 2000, Windows XP and Windows Server 2003 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 – Begin with IBM®, SQL or SYS. – Include accented characters.

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.
3. Trailing blanks from user IDs and passwords are removed.

Related concepts:

- "General naming rules" on page 291
- "Federated database object naming rules" on page 294

Federated database object naming rules

Table 21. 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)

Related concepts:

- "General naming rules" on page 291

Additional restrictions and recommendations regarding the use of schema names

- 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.

Related concepts:

- "General naming rules" on page 291

Maintaining passwords on servers

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® Universal Database (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 UDB Configuration Assistant (CA) can also be used to change the password.

Related concepts:

- "General naming rules" on page 291
- "DB2 UDB object naming rules" on page 291
- "Workstation naming rules" on page 296
- "User, user ID and group naming rules" on page 294
- "Federated database object naming rules" on page 294
- "Delimited identifiers and object names" on page 293
- "Additional restrictions and recommendations regarding the use of schema names" on page 295

Workstation naming rules

A *workstation name* specifies the NetBIOS name for a database server, database client, or DB2® Universal Database (DB2 UDB) Personal Edition that resides on the local workstation. This name is stored in the database manager configuration file. The workstation name is known as the *workstation nname*.

In addition, the name you specify:

- Can contain 1 to 8 characters
- Cannot include &, #, or @
- Must be unique within the network

In a partitioned database system, there is still only one workstation *nname* that represents the entire partitioned database system, but each node has its own derived unique NetBIOS *nname*.

The workstation *nname* that represents the partitioned database system is stored in the database manager configuration file for the database partition server that owns the instance.

Each node's unique *nname* is a derived combination of the workstation *nname* and the node number.

If a node does not own an instance, its NetBIOS *nname* is derived as follows:

1. The first character of the instance-owning machine's workstation *nname* is used as the first character of the node's NetBIOS *nname*.
2. The next 1 to 3 characters represent the node number. The range is from 1 to 999.
3. The remaining characters are taken from the instance-owning machine's workstation *nname*. The number of remaining characters depends on the length of the instance-owning machine's workstation *nname*. This number can be from 0 to 4.

For example:

Instance-Owning Machine's *Workstation nname*	**Node Number**	**Derived Node NetBIOS** *nname*
GEORGE	3	G3ORGE
A	7	A7
B2	94	B942
N0076543	21	N216543
GEORGE5	1	G1RGE5

If you have changed the default workstation *nname* during the installation, the workstation *nname*'s last 4 characters should be unique across the NetBIOS network to minimize the chance of deriving a conflicting NetBIOS *nname*.

Related concepts:

- "General naming rules" on page 291

Naming rules in an NLS environment

The basic character set that can be used in database names consists of the single-byte uppercase and lowercase Latin letters (A...Z, a...z), the Arabic numerals (0...9) and the underscore character (_). This list is augmented with three special characters (#, @, and $) to provide compatibility with host database products. Use special characters #, @, and $ with care in an NLS environment because they are not included in the NLS host (EBCDIC) invariant character set. Characters from the extended character set can also be used, depending on the code page that is being used. If you are using the database in a multiple code page environment, you must ensure that all code pages support any elements from the extended character set you plan to use.

When naming database objects (such as tables and views), program labels, host variables, cursors, and elements from the extended character set (for example, letters with diacritical marks) can also be used. Precisely which characters are available depends on the code page in use.

Extended Character Set Definition for DBCS Identifiers:

In DBCS environments, the extended character set consists of all the characters in the basic character set, plus the following:

- All double-byte characters in each DBCS code page, except the double-byte space, are valid letters.
- The double-byte space is a special character.
- The single-byte characters available in each mixed code page are assigned to various categories as follows:

Category	Valid Code Points within each Mixed Code Page
Digits	x30-39
Letters	x23-24, x40-5A, x61-7A, xA6-DF (A6-DF for code pages 932 and 942 only)
Special Characters	All other valid single-byte character code points

Related concepts:

- "General naming rules" on page 291
- "DB2 UDB object naming rules" on page 291
- "Workstation naming rules" on page 296

Naming rules in a Unicode environment

In a UCS-2 database, all identifiers are in multibyte UTF-8. Therefore, it is possible to use any UCS-2 character in identifiers where the use of a character in the extended character set (for example, an accented character, or a multibyte character) is allowed by DB2® Universal Database (DB2 UDB).

Clients can enter any character that is supported by their environment, and all the characters in the identifiers will be converted to UTF-8 by the database manager. Two points must be taken into account when specifying national language characters in identifiers for a UCS-2 database:

- Each non-ASCII character requires two to four bytes. Therefore, an *n*-byte identifier can only hold somewhere between *n*/4 and *n* characters, depending on the ratio of ASCII to non-ASCII characters. If you have only one or two non-ASCII (for example, accented) characters, the limit is closer to *n* characters, while for an identifier that is completely non-ASCII (for example, in Japanese), only *n*/4 to *n*/3 characters can be used.
- If identifiers are to be entered from different client environments, they should be defined using the common subset of characters available to those clients. For example, if a UCS-2 database is to be accessed from Latin-1, Arabic, and Japanese environments, all identifiers should realistically be limited to ASCII.

Related concepts:

- "General naming rules" on page 291
- "DB2 UDB object naming rules" on page 291
- "Workstation naming rules" on page 296

Appendix B. Using automatic client rerouting

This appendix describes how the automatic client rerouting works and how to make sure that it works correcty in your environment.

Automatic client reroute description and setup

Whenever a server crashes, each client that is connected to that server receives a communication error which terminates the connection resulting in an application error. In cases where availability is important, you should have implemented either a redundant setup or the ability to fail the server over to a standby node. In either case, the DB2® Universal Database (DB2 UDB) client code attempts to re-establish the connection to the original server which may be running on a failover node (the IP address fails over as well), or to a new server.

When the connection is re-established, the application receives an error message that informs it of the transaction failure, but the application can continue with the next transaction.

The main goal of the automatic client reroute feature is to enable a DB2 UDB client application to recover from a loss of communications so that the application can continue its work with minimal interruption. As the name applies, rerouting is central to the support of continuous operations. But rerouting is only possible when there is an alternate location that is identified to the client conection.

The automatic client reroute feature could be used within the following configurable environments:

1. Enterprise Server Edition (ESE) with the data partitioning feature (DPF)
2. Data Propagator (DPROPR)-style replication
3. High availability cluster multiprocessor (HACMP)
4. High availability disaster recovery (HADR).

   Automatic client reroute works in conjunction with HADR to allow a client application to continue its work with minimal interruption after a failover of the database being accessed.

In order for DB2 UDB to have the ability to recover from a loss of communications, an alternative server location must be specified before the loss of communication occurs. You can specify the alternate server by using the UPDATE ALTERNATE SERVER FOR DATABASE command. In order to ensure the alternate server location specified applies to all clients, the alternate server location has to be specified at the server side. The alternate server is ignored if it is set at the client instance.

For example, assume a database is located at the node called “N1” (with a hostname of XXX and a port number YYY). The database administrator would like to set the alternate server location to be at the hostname = AAA with a port number of 123. Here is the command the database administrator would run at node N1 (on the server instance):

```
db2 update alternate server for database db2 using hostname AAA port 123
```

After the database administrator specifies the alternate server location on a particular database at the server instance, the alternate server location is returned back to the client at connect time. If communication is lost for any reason, DB2 UDB client code will be able to re-establish the connection by using the alternate server information that had been returned from the server. If the alternate server is not specified at the server, automatic client reroute will not be processed.

In general, if an alternate server is specified, automatic client reroute will be enabled when a communication error (sqlcode -30081) or a sqlcode -1224 is detected. However, in a high availability disaster recovery (HADR) environment, it will also be enabled if sqlcode -1776 is returned back from the HADR standby server.

Related concepts:

• "Automatic client reroute limitations" on page 300

Related reference:

• "Automatic client reroute examples" on page 301

Automatic client reroute limitations

There are some limitations with use of the automatic client reroute feature:

- Automatic client reroute is only supported when the communications protocol used for connecting to the DB2® Universal Database (DB2 UDB) server, or to the DB2 Connect™ server, is TCP/IP. This means that if the connection is using a different protocol other than TCP/IP, the automatic client reroute feature will not be enabled. Even if DB2 UDB is setup for a loopback, TCP/IP communications protocol must be used in order accommodate the automatic client reroute feature.
- If the connection is reestablished to the alternate server location, any new connection to the same database alias will be connected to the alternate server location. If you want any new connection to be established, to the original location in case the problem on the original location is fixed, there are a couple of options from which to choose:
  - You need to take the alternate server offline and allow the connections to fail back over to the original server. (This assumes that the original server has been cataloged using the UPDATE ALTERNATE SERVER command such that it is set to be the alternate location for the alternate server.)
  - You could catalog a new database alias to be used by the new connections.
  - You could uncatalog the database entry and re-catalog it again.
- DB2 UDB for Linux, UNIX®, and Windows® operating systems will support the automatic client reroute feature in both the client and the server side. Other DB2 UDB families do not currently support this feature.
- A datasharing SYSPLEX setup on DB2 UDB for z/OS™ may return a list of servers that is available to be connected. However, the list is only kept in memory, when a communication failure happens. At that time, the DB2 UDB client will use the list to determine the location of the appropriate alternate server for connection.
- The DB2 UDB server installed in the alternate host server must be the same version (but could have a higher FixPak) when compared to the DB2 UDB installed on the original host server.
- Regardless of whether you have authority to update the database directory at the client machine, the alternate server information is always kept in memory. In

other words, if you did not have authority to update the database directory (or because it is a read-only database directory), other applications will not be able to determine and use the alternate server, because the memory is not shared among applications.

- The same authentication is applied to all alternate locations. This means that the client will be unable to reestablish the database connection if the alternate location has a different authentication type than the original location.
- When there is a communication failure, all session resources such as global temporary tables, identity, sequences, cursors, server options (SET SERVER OPTION) for federated processing and special registers are all lost. The application is responsible to re-establish the session resources in order to continue processing the work. You do not have to run any of the special register statements after the connection is re-established, because DB2 UDB will re-play the special register statements that were issued before the communication error. However, some of the special registers will not be replayed. They are:
  - SET ENCRYPTPW
  - SET EVENT MONITOR STATE
  - SET SESSION AUTHORIZATION
  - SET TRANSFORM GROUP

  Note: If the client is using CLI, JCC Type 2 or Type 4 drivers, after the connection is re-established, then for those SQL statements that have been prepared against the original server, they are implicitly re-prepared with the new server. However, for embedded SQL routines (for example, SQC or SQX applications), they will not be re-prepared.

An alternate way to do Automatic Client Reroute is to use the DNS entry to specify an alterate IP address for a DNS entry. The idea is to specify a second IP address (an alternate server location) in the DNS entry, the client would not know about an alternate server but at connect time DB2 UDB would alternate between the IP addresses for the DNS entry.

Related concepts:

- "Automatic client reroute implementation" on page 70
- "Automatic client reroute description and setup" on page 299

Related tasks:

- "Identify an alternate server for a database" on page 67

Related reference:

- "Automatic client reroute examples" on page 301

Automatic client reroute examples

Here is an automatic client reroute example for a client application (shown using pseudo-code only):

```
int checkpoint = 0;

check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
   if (sqlca->sqlcode == -30081)
   {
      // as communication is lost, terminate the application right away
      exit(1);
```

```
   }
   else
   {
      // print out the error
      printf(...);

 if (sqlca->sqlcode == -30108)
 {
    // connection is re-established, re-execute the failed transaction
         if (checkpoint == 0)
         {
             goto checkpt0;
         }
    else if (checkpoint == 1)
         {
            goto checkpt1;
         }
         else if (checkpoint == 2)
         {
             goto checkpt2;
         }
         ....
         exit;
 }
   }
}

main()
{
   connect to mydb;
   check_sqlca("connect failed", &sqlca);

checkpt0:
   EXEC SQL set current schema XXX;
   check_sqlca("set current schema XXX failed", &sqlca);

   EXEC SQL create table t1...;
   check_sqlca("create table t1 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 1;
   }

checkpt1:
   EXEC SQL set current schema YYY;
   check_sqlca("set current schema YYY failed", &sqlca);

   EXEC SQL create table t2...;
   check_sqlca("create table t2 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);

   if (sqlca.sqlcode == 0)
   {
      checkpoint = 2;
   }
...
}
```

At the client machine, the database called "mydb" is cataloged which references a node "hornet" where "hornet" is also cataloged in the node directory (hostname "hornet" with port number 456).

Example 1 (involving a non-HADR database)

At the server "hornet" (hostname equals hornet with a port number), a database "mydb" is created. Furthermore, the database "mydb" is also created at the alternate server (hostname "montero" with port number 456). You will also need to update the alternate server for database "mydb" at server "hornet" as follows:

```
db2 update alternate server for database mydb using hostname montero port 456
```

In the sample application above, and without having the automatic client reroute feature set up, if there is a communication error in the `create table t1` statement, the application will be terminated. With the automatic client reroute feature set up, DB2 UDB will try to establish the connection to host "hornet" (with port 456) again. If it is still not working, DB2 UDB will try the alternate server location (host "montero" with port 456). Assuming there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements (and to re-run the failed transaction).

Example 2 (involving an HADR database)

At the server "hornet" (hostname equals hornet with a port number), primary database "mydb" is created. A secondary database is also created at host "montero" with port 456. Information on how to setup HADR for both a primary and secondary database is found in *Data Recovery and High Availability Guide and Reference*. You will also need to update the alternate server for database "mydb" as follows:

```
db2 update alternate server for database mydb using hostname montero port 456
```

In the sample application above, and without having the automatic client reroute feature set up, if there is a communication error in the `create table t1` statement, the application will be terminated. With the automatic client reroute feature set up, DB2 UDB will try to establish the connection to host "hornet" (with port 456) again. If it is still not working, DB2 UDB will try the alternate server location (host "montero" with port 456). Assuming there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements (and to re-run the failed transaction).

Related concepts:

- "Automatic client reroute description and setup" on page 299

Related tasks:

- "Identify an alternate server for a database" on page 67

Appendix C. Using lightweight directory access protocol (LDAP) directory services

Introduction to Lightweight Directory Access Protocol (LDAP)

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 needs to search the LDAP directory server once. Once the information is retrieved from the LDAP directory server, it is stored or cached on the local machine based on the values of the *dir_cache* database manager configuration parameter and the DB2LDAPCACHE registry variable. The *dir_cache* database manager configuration parameter is used to store database, node, and DCS directory files in a memory cache. The directory cache is used by an application until the application closes. The DB2LDAPCACHE registry variable is used to store database, node, and DCS directory files in a local disk cache.

- 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, then read the information from LDAP once and cache it into the local database, node, and DCS directories.

Note: The DB2LDAPCACHE registry variable is only applicable to the database and node directories.

Related concepts:

- "Lightweight Directory Access Protocol (LDAP) Directory Service" on page 68
- "Support for Active Directory" on page 307
- "LDAP support and DB2 Connect" on page 321
- "Security considerations in an LDAP environment" on page 321
- "Security considerations for Active Directory" on page 322
- "DB2 registry and environment variables" in the *Administration Guide: Performance*
- "Extending the LDAP directory schema with DB2 object classes and attributes" on page 323

Related tasks:

- "Configuring DB2 to use Active Directory" on page 308
- "Configuring DB2 in the IBM LDAP environment" on page 308

Related reference:

Supported LDAP client and server configurations

The following table summarizes the supported LDAP client and server configurations:

Table 22. Supported LDAP Client and Server Configurations

	IBM SecureWay Directory	Microsoft Active Directory	Netscape LDAP server
IBM LDAP Client	Supported	Supported	Supported
Microsoft LDAP/ADSI Client	Supported	Supported	Supported

IBM SecureWay Directory Version 3.1 is an LDAP Version 3 server available for Windows NT, AIX, Solaris Operating Environment, and HP-UX. SecureWay directory is shipped as part of the base operating system on AIX and iSeries (AS/400), and with OS/390 Security Server.

The 32-bit version of DB2 Universal Database™ (DB2 UDB) supports IBM LDAP client on AIX, Solaris Operating Environment, HP-UX 11.11, Windows, Linux IA32 and Linux/390.

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 is included with the Windows operating system.

When running on Windows operting systems, DB2 supports using either the IBM LDAP client or the Microsoft LDAP client to access the IBM SecureWay Directory Server. To explicitly select the IBM LDAP client, use the **db2set** command to set the DB2LDAP_CLIENT_PROVIDER registry variable to "IBM".

Related concepts:

- "Introduction to Lightweight Directory Access Protocol (LDAP)" on page 305
- "Support for Active Directory" on page 307

Support for Active Directory

DB2® Universal Database (DB2 UDB) 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 UDB 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.

Property pages for the ibm_db2Node and ibm_db2Database objects can be viewed or modified using the *Active Directory Users and Computer* Management Console (MMC) at a domain controller. To setup the property page, run the regsrv32 command to register the property pages for the DB2 objects as follows:

```
regsvr32 %DB2PATH%\bin\db2ads.dll
```

You can view the objects by using the *Active Directory Users and Computer* Management Console (MMC) at a domain controller. To get to this administration tool, follow Start—> Program—> Administration Tools—> Active Directory Users and Computer.

Note: You must select *Users, Groups, and Computers as containers* from the View menu to display the DB2 UDB objects under the computer objects.

Note: If DB2 UDB is not installed on the domain controller, you can still view the property pages of DB2 UDB objects by copying the db2ads.dll file from %DB2PATH%\bin and the resource DLL db2adsr.dll from %DB2PATH%\msg\locale-name to a local directory on the domain controller. (The directory where you place these two copied files must be one of those found in the PATH user/system environment variable.) Then, you run the regsvr32 command from the local directory to register the DLL.

Related concepts:

- "Security considerations for Active Directory" on page 322

Related tasks:

- "Configuring DB2 to use Active Directory" on page 308
- "Extending the directory schema for Active Directory" on page 323

Related reference:

- "DB2 objects in the Active Directory" on page 325

Configuring DB2 to use Active Directory

Procedure:

In order to access Microsoft Active Directory, ensure that the following conditions are met:

1. The machine that runs DB2 Universal Database™ (DB2 UDB) must belong to a Windows 2000 or Windows Server 2003 domain.
2. The Microsoft® LDAP client is installed. The Microsoft® LDAP client is part of the Windows 2000, Windows XP, and Windows Server 2003 operating systems. For Windows 98, Windows NT, or Windows Me, you need to verify that the Active Directory client extension (`wldap32.dll`) exists under the system directory.
3. Enable the LDAP support. For Windows 2000, Windows XP, or Windows Server 2003, the LDAP support is enabled by the installation program. For Windows 98, Windows NT, or Windows Me, 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 UDB to read information from the Active Directory.

Related concepts:

- "Support for Active Directory" on page 307
- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related tasks:

- "Configuring the LDAP user for DB2 applications" on page 310

Configuring DB2 in the IBM LDAP environment

Procedure:

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 or Windows NT, you must explicitly enable LDAP by setting the DB2_ENABLE_LDAP registry variable to "YES" using the **db2set** command. The default LDAP client to use on all Windows operating systems is Microsoft's. If you want to use the IBM LDAP client, you must set the DB2LDAP_CLIENT_PROVIDER registry variable to "IBM", 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.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related tasks:

- "Configuring DB2 to use Active Directory" on page 308
- "Creating an LDAP user" on page 309

Related reference:

- "db2set - DB2 Profile Registry Command" in the *Command Reference*

Creating an LDAP user

Procedure:

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.

Related tasks:

- "Configuring DB2 in the IBM LDAP environment" on page 308
- "Configuring the LDAP user for DB2 applications" on page 310

Configuring the LDAP user for DB2 applications

Procedure:

When using the Microsoft LDAP client, the LDAP user is the same as the operating system user account. However, when working with the IBM LDAP client and before using 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
```

Related tasks:

- "Creating an LDAP user" on page 309

Related reference:

- "db2ldcfg - Configure LDAP Environment Command" in the *Command Reference*

Registration of DB2 servers after installation

Procedure:

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 Enterprise Server Edition 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 Enterprise Server Edition 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.

Related concepts:

- "rah and db2_all commands overview" on page 343

Related tasks:

- "Update the protocol information for the DB2 server" on page 312
- "Catalog a node alias for ATTACH" on page 313
- "Deregistering the DB2 server" on page 314
- "Attaching to a remote server in the LDAP environment" on page 315

Related reference:

- "REGISTER Command" in the *Command Reference*
- "CATALOG LDAP NODE Command" in the *Command Reference*

Update the protocol information for the DB2 server

Procedure:

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.
- 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>
```

Related tasks:

- "Registration of DB2 servers after installation" on page 310
- "Catalog a node alias for ATTACH" on page 313
- "Attaching to a remote server in the LDAP environment" on page 315

Related reference:

- "UPDATE LDAP NODE Command" in the *Command Reference*

Rerouting LDAP clients to another server

Just as with the ability to reroute clients on a system failure, the same ability is also available to you when working with LDAP.

Prerequisites:

The DB2_ENABLE_LDAP registry variable is set to "Yes".

Procedure:

Within an LDAP environment, all database and node directory information is maintained at an LDAP server. The client retrieves information from the LDAP directory. This information is updated in its local database and node directories if the DB2LDAPCACHE registry variable is set to "Yes".

Use the UPDATE ALTERNATE SERVER FOR LDAP DATABASE command to define the alternate server for a database that represents the DB2 Universal Database™ (DB2 UDB) in LDAP.

Once established, this alternate server information is returned to the client upon connection.

Note: When you enter UPDATE ALTERNATE SERVER FOR DATABASE command (notice that it is not "FOR LDAP DATABASE") at the server instance, and if LDAP support is enabled (DB2_ENABLE_LDAP=Yes) on the server, and if the LDAP user ID and password is cached (**db2ldcfg** was previously run), then the alternate server for the database is automatically, or implicitly, updated on the LDAP server. This works as if you entered UPDATE ALTERNATE SERVER FOR LDAP DATABASE explicitly.

Related concepts:

- "Automatic client reroute implementation" on page 70
- "Automatic client reroute description and setup" on page 299

Catalog a node alias for ATTACH

Procedure:

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>
```

Related tasks:

- "Registration of DB2 servers after installation" on page 310

- "Attaching to a remote server in the LDAP environment" on page 315

Related reference:
- "CATALOG LDAP NODE Command" in the *Command Reference*
- "UNCATALOG LDAP NODE Command" in the *Command Reference*

Deregistering the DB2 server

Procedure:

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.

Related tasks:
- "Registration of DB2 servers after installation" on page 310

Related reference:
- "DEREGISTER Command" in the *Command Reference*

Registration of databases in the LDAP directory

Procedure:

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"
```

Related tasks:
- "Registration of DB2 servers after installation" on page 310

- "Deregistering the database from the LDAP directory" on page 316

Related reference:

- "CATALOG LDAP DATABASE Command" in the *Command Reference*

Attaching to a remote server in the LDAP environment

Procedure:

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, the database manager 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, the database manager 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 machanism exists so that the client only searches the LDAP server once. Once the information is retrieved, it is stored or cached on the local machine 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, then read the information from LDAP server once and cache it into the local database, node, and DCS directories.

Note: The caching of LDAP information is not applicable to user-level CLI or DB2 profile registry variables.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related tasks:

- "Registration of DB2 servers after installation" on page 310
- "Update the protocol information for the DB2 server" on page 312
- "Catalog a node alias for ATTACH" on page 313
- "Registration of databases in the LDAP directory" on page 314

Related reference:

- "ATTACH Command" in the *Command Reference*

Deregistering the database from the LDAP directory

Procedure:

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>
```

Related tasks:

- "Registration of databases in the LDAP directory" on page 314

Related reference:

- "UNCATALOG LDAP DATABASE Command" in the *Command Reference*

Refreshing LDAP entries in local database and node directories

Procedure:

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 machanism exists so that the client only searches the LDAP server once. Once the information is retrieved, it is stored or cached on the local machine 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, then read the information from LDAP server once and cache it into the local database, node, and DCS directories.

Note: The caching of LDAP information is not applicable to user-level CLI or DB2 profile registry variables.

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.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related reference:

- "dir_cache - Directory cache support configuration parameter" in the *Administration Guide: Performance*
- "REFRESH LDAP Command" in the *Command Reference*

Searching the LDAP directory partitions or domains

Procedure:

DB2 UDB 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 UDB 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 UDB profile registry variables DB2LDAP_KEEP_CONNECTION and DB2LDAP_SEARCH_SCOPE are the only registry variables that support setting the variable at the global level in LDAP.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related tasks:

• "Declaring registry and environment variables" on page 28

Registering host databases in LDAP

Procedure:

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.

If LDAP support is available at the DB2® Connect gateway, and the database is not found at the gateway database directory, DB2 UDB will look up LDAP and attempt to keep the found information.

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”.

Related reference:

- “REGISTER Command” in the *Command Reference*
- “CATALOG LDAP DATABASE Command” in the *Command Reference*

Setting DB2 registry variables at the user level in the LDAP environment

Procedure:

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 Operating Environment.

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
  ```

Related tasks:

- “Declaring registry and environment variables” on page 28

Related reference:

- “db2set - DB2 Profile Registry Command” in the *Command Reference*

Enabling LDAP support after installation is complete

Procedure:

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, Windows 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.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related tasks:

- "Disabling LDAP support" on page 321

Related reference:

- "REGISTER Command" in the *Command Reference*
- "db2set - DB2 Profile Registry Command" in the *Command Reference*
- "CATALOG LDAP DATABASE Command" in the *Command Reference*

Disabling LDAP support

Procedure:

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".

Related tasks:

- "Declaring registry and environment variables" on page 28
- "Enabling LDAP support after installation is complete" on page 319

Related reference:

- "DEREGISTER Command" in the *Command Reference*

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.

Related concepts:

- "Introduction to Lightweight Directory Access Protocol (LDAP)" on page 305
- "Security considerations in an LDAP environment" on page 321

Security considerations in an LDAP environment

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.

Related concepts:

- "Security considerations for Active Directory" on page 322

Security considerations for 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 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"

Related concepts:

- "Security considerations in an LDAP environment" on page 321

Extending the LDAP 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 UDB Version 8.1 and earlier are included in the base schema. In this case, you do not have to extend the base schema with DB2 object classes and attributes. However, there are two new attributes for DB2 UDB Version 8.2 that are not included in the base schema. In this case, you have to extend the base schema with the two new DB2 UDB attributes.

Related concepts:

- "Extending the directory schema for IBM SecureWay Directory Server" on page 327

Related tasks:

- "Extending the directory schema for Active Directory" on page 323

Extending the directory schema for Active Directory

Procedure:

Before DB2 Universal Database™ (DB2 UDB) can store information in the Active Directory, the directory schema needs to be extended to include the new DB2 UDB 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 UDB Schema Installation program, **db2schex** before the first installation of DB2 UDB on any machine that is part of a Windows domain.

The **db2schex** program is included on the product CD-ROM. The location of this program on the CD-ROM is under the db2 directory, the windows subdirectory, and the utilities subdirectory. For example:

```
x:\db2\windows\utilities\
```

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.

You need to run the **db2schex.exe** command that comes with the DB2 UDB Version 8.2 product to extend the directory schema.

If you have run the **db2schex.exe** command that came with the previous version of DB2 for Windows product, when you run this same command again that come with DB2 UDB Version 8.2, it will add the following two optional attributes to the ibm-db2Database class:

```
ibm-db2AltGwPtr
ibm-db2NodePtr
```

If you have not run the **db2schex.exe** command that came with the previous version of DB2 UDB for Windows product, when you run this same command that come with DB2 Version 8.2, it will add all the classes and attributes for DB2 UDB LDAP support.

Examples:

- To install the DB2 UDB schema:

  ```
  db2schex
  ```
- To install the DB2 UDB 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 UDB schema:

  ```
  db2schex -u
  ```

- To uninstall the DB2 UDB schema and ignore errors:

```
db2schex -u -k
```

The DB2 UDB 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.

Related concepts:

- "Extending the LDAP directory schema with DB2 object classes and attributes" on page 323

DB2 objects in the 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.

Related reference:

- "LDAP object classes and attributes used by DB2" on page 331

Netscape LDAP directory support and attribute definitions

The supported level for Netscape LDAP Server is v4.12 or later.

Within Netscape LDAP Server Version 4.12 or later, the Netscape Directory Server allows application to extend the schema by adding attribute and object class definitions into the following two files, `slapd.user_oc.conf` and `slapd.user_at.conf`. These two files are located in the

```
<Netscape_install path>\slapd-<machine_name>\config
```

directory.

Note: If you are using iPlan Directory Server 5.0, then you must review the documentation that accompanies that product for detailed instructions on how to extend the schema.

The DB2 attributes must be added to the slapd.user_at.conf as follows:

```
#########################################################################
#
# IBM DB2 Universal Database
# Attribute Definitions
#
# bin -> binary
# ces -> case exact string
# cis -> case insensitive string
# dn -> distinguished name
#
#########################################################################

attribute binProperty                    1.3.18.0.2.4.305     bin
attribute binPropertyType                1.3.18.0.2.4.306     cis
attribute cesProperty                    1.3.18.0.2.4.307     ces
attribute cesPropertyType                1.3.18.0.2.4.308     cis
attribute cisProperty                    1.3.18.0.2.4.309     cis
attribute cisPropertyType                1.3.18.0.2.4.310     cis
attribute propertyType                   1.3.18.0.2.4.320     cis
attribute systemName                     1.3.18.0.2.4.329     cis
attribute db2nodeName                    1.3.18.0.2.4.419     cis
attribute db2nodeAlias                   1.3.18.0.2.4.420     cis
attribute db2instanceName                1.3.18.0.2.4.428     cis
attribute db2Type                        1.3.18.0.2.4.418     cis
attribute db2databaseName                1.3.18.0.2.4.421     cis
attribute db2databaseAlias               1.3.18.0.2.4.422     cis
attribute db2nodePtr                     1.3.18.0.2.4.423     dn
attribute db2gwPtr                       1.3.18.0.2.4.424     dn
attribute db2additionalParameters        1.3.18.0.2.4.426     cis
attribute db2ARLibrary                   1.3.18.0.2.4.427     cis
attribute db2authenticationLocation      1.3.18.0.2.4.425     cis
attribute db2databaseRelease             1.3.18.0.2.4.429     cis
attribute DCEPrincipalName               1.3.18.0.2.4.443     cis
```

The DB2 object classes must be added to the slapd.user_oc.conf file as follows:

```
#########################################################################
#
# IBM DB2 Universal Database
# Object Class Definitions
#
#########################################################################

objectclass eProperty
        oid 1.3.18.0.2.6.90
        requires
                objectClass
        allows
                cn,
                propertyType,
                binProperty,
                binPropertyType,
                cesProperty,
                cesPropertyType,
                cisProperty,
                cisPropertyType

objectclass eApplicationSystem
        oid 1.3.18.0.2.6.84
        requires
                objectClass,
                systemName


objectclass DB2Node
        oid 1.3.18.0.2.6.116
```

```
                requires
                        objectClass,
                        db2nodeName
                allows
                        db2nodeAlias,
                        host,
                        db2instanceName,
                        db2Type,
                        description,
                        protocolInformation

        objectclass DB2Database
                oid 1.3.18.0.2.6.117
                requires
                        objectClass,
                        db2databaseName,
                        db2nodePtr
                allows
                        db2databaseAlias,
                        description,
                        db2gwPtr,
                        db2additionalParameters,
                        db2authenticationLocation,
                        DCEPrincipalName,
                        db2databaseRelease,
                        db2ARLibrary
```

After adding the DB2 schema definition, the Directory Server must be restarted for all changes to be active.

Related reference:

• "LDAP object classes and attributes used by DB2" on page 331

Extending the directory schema for IBM SecureWay Directory Server

If you are using IBM® SecureWay® LDAP Directory Server version 3.1 or later, all the object classes and attributes that are required by DB2® Universal Database (DB2 UDB) before Version 8.2 are included in the base schema. Run the following to extend the base schema with new DB2 UDB attributes introduced in Version 8.2:

```
ldapmodify -c -h <machine_name>:389 -D <dn> -w <password> -f altgwnode.ldif
```

The following is the content of the `altgwnode.ldif` file:

```
dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3092
 NAME 'db2altgwPtr'
 DESC 'DN pointer to DB2 alternate gateway (node) object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3092
 DBNAME ('db2altgwPtr' 'db2altgwPtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3093
 NAME 'db2altnodePtr'
 DESC 'DN pointer to DB2 alternate node object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3093
 DBNAME ('db2altnodePtr' 'db2altnodePtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:            cn=schema
changetype:    modify
replace:       objectclasses
objectclasses: (
 1.3.18.0.2.6.117
 NAME 'DB2Database'
 DESC 'DB2 database'
 SUP cimSetting
 MUST ( db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName ) )
```

The `altgwnode.ldif` and `altgwnode.readme`files can be found at URL: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

After adding the DB2 schema definition, the Directory Server must be restarted for all changes to be active.

Related concepts:

- "Extending the LDAP directory schema with DB2 object classes and attributes" on page 323
- "Extending the directory schema for Sun One Directory Server" on page 329

Related tasks:

- "Extending the directory schema for Active Directory" on page 323

Extending the directory schema for Sun One Directory Server

The Sun One Directory Server is also known as the Netscape or iPlanet directory server.

To have the Sun One Directory Server work in your environment, add the `60ibmdb2.ldif` file to the following directory:

On Windows®, if you have iPlanet installed in `C:\iPlanet\Servers`, add the above file to `.\sldap-<machine_name>\config\schema`.

On UNIX®, if you have iPlanet installed in `/usr/iplanet/servers`, add the above file to `./slapd-<machine_name>/config/schema`.

The following is the contents of the file:

```
########################################################################
# IBM&regtm; DB2&regtm; Universal Database
########################################################################
dn: cn=schema
########################################################################
# Attribute Definitions (Before V8.2)
########################################################################
attributetypes: ( 1.3.18.0.2.4.305  NAME 'binProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.5               X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.306  NAME 'binPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.307  NAME 'cesProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.308  NAME 'cesPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.309  NAME 'cisProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.310  NAME 'cisPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.320  NAME 'propertyType'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.329  NAME 'systemName'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.419  NAME 'db2nodeName'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.420  NAME 'db2nodeAlias'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.428  NAME 'db2instanceName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.418  NAME 'db2Type'                   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.421  NAME 'db2databaseName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.422  NAME 'db2databaseAlias'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.426  NAME 'db2additionalParameters'   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.427  NAME 'db2ARLibrary'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.425  NAME 'db2authenticationLocation' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.429  NAME 'db2databaseRelease'        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.443  NAME 'DCEPrincipalName'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.423  NAME 'db2nodePtr'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.424  NAME 'db2gwPtr'                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
########################################################################
# Attribute Definitions (V8.2)
########################################################################
attributetypes: ( 1.3.18.0.2.4.3092 NAME 'db2altgwPtr'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.3093 NAME 'db2altnodePtr'             SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
########################################################################
# Object Class Definitions
# DB2Database for V8.2 has the above two new optional attributes.
########################################################################
objectClasses: ( 1.3.18.0.2.6.90  NAME 'eProperty'          SUP top STRUCTURAL
 MAY ( cn $ propertyType $ binProperty $ binPropertyType $ cesProperty $ cesPropertyType $ cisProperty $ cisPropertyType )
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.84  NAME 'eApplicationSystem' SUP top STRUCTURAL MUST systemName
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.116 NAME 'DB2Node'            SUP top STRUCTURAL MUST db2nodeName
 MAY ( db2instanceName $ db2nodeAlias $ db2Type $ description $ host $ protocolInformation )
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.117 NAME 'DB2Database'        SUP top STRUCTURAL MUST (db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName $ description )
 X-ORIGIN 'IBM DB2' )
```

The `60ibmdb2.ldif` and `60ibmdb2.readme`files can be found at URL: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

After adding the DB2 schema definition, the Directory Server must be restarted for all changes to be active.

Related concepts:

- "Extending the LDAP directory schema with DB2 object classes and attributes" on page 323
- "Extending the directory schema for IBM SecureWay Directory Server" on page 327

Related tasks:

- "Extending the directory schema for Active Directory" on page 323

LDAP object classes and attributes used by DB2

The following tables describe the object classes that are used by DB2:

Table 23. 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 24. 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
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 25. 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 26. 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
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

Table 26. DB2Node (continued)

Class	DB2Node
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 UDB for Windows client, 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 UDB client tries to use this information to configure the SNA stack if it knows how to configure the stack.

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 27. DB2Database

Class	**DB2Database**
Active Directory LDAP Display Name	ibm-db2Database
Active Directory Common Name (cn)	ibm-db2Database
Description	Represents a DB2 database
SubClassOf	top
Required Attribute(s)	db2databaseName db2nodePtr
Optional Attribute(s)	db2databaseAlias db2additionalParameter db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName db2altgwPtr db2altnodePtr
Type	structural
OID (object identifier)	1.3.18.0.2.6.117
GUID (Global Unique Identifier)	b3afd659-5c5b-11d3-b818-002035559151

Table 28. 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 29. db2authenticationLocation

Attribute	**db2authenticationLocation**
Active Directory LDAP Display Name	ibm-db2AuthenticationLocation
Active Directory Common Name (cn)	ibm-db2AuthenticationLocation
Description	Specifies where authentication takes place

Table 29. db2authenticationLocation (continued)

Attribute	db2authenticationLocation
Syntax	Case Ignore String
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 30. 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 31. 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 32. db2databaseName

Attribute	db2databaseName
Active Directory LDAP Display Name	ibm-db2DatabaseName
Active Directory Common Name (cn)	ibm-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 33. 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 34. 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 35. db2nodeName

Attribute	**db2nodeName**
Active Directory LDAP Display Name	ibm-db2NodeName
Active Directory Common Name (cn)	ibm-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 36. 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

Table 36. db2nodePtr (continued)

Attribute	db2nodePtr
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 37. db2altnodePtr

Attribute	db2altnodePtr
Active Directory LDAP Display Name	ibm-db2AltNodePtr
Active Directory Common Name (cn)	ibm-db2AltNodePtr
Description	Pointer to the Node (DB2Node) object that represents the alternate database server
Syntax	Distinguished Name
Maximum Length	1000
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.3093
GUID (Global Unique Identifier)	5694e266-2059-4e32-971e-0778909e0e72

Table 38. 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
GUID (Global Unique Identifier)	b3afd31b-5c5b-11d3-b818-002035559151

Table 39. db2altgwPtr

Attribute	db2altgwPtr
Active Directory LDAP Display Name	ibm-db2AltGwPtr
Active Directory Common Name (cn)	ibm-db2AltGwPtr
Description	Pointer to the Node object that represents the alternate gateway server
Syntax	Distinguished Name
Maximum Length	1000
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.3092
GUID (Global Unique Identifier)	70ab425d-65cc-4d7f-91d8-084888b3a6db

Table 40. 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 41. 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 42. 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
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.443
GUID (Global Unique Identifier)	b3afd32d-5c5b-11d3-b818-002035559151

Table 43. 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.

Table 43. cesProperty (continued)

Attribute	cesProperty
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 44. 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 45. 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 46. cisPropertyType

Attribute	cisPropertyType
Active Directory LDAP Display Name	ibm-cisPropertyType
Active Directory Common Name (cn)	ibm-cisPropertyType

Table 46. cisPropertyType (continued)

Attribute	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 47. binProperty

Attribute	binProperty
Active Directory LDAP Display Name	ibm-binProperty
Active Directory Common Name (cn)	ibm-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 48. 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 49. 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
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.320
GUID (Global Unique Identifier)	b3afd4ed-5c5b-11d3-b818-002035559151

Table 50. 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

Related concepts:

• "Introduction to Lightweight Directory Access Protocol (LDAP)" on page 305

Appendix D. Issuing commands to multiple database partitions

Issuing commands in a partitioned database environment

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**.

Related concepts:

- "rah and db2_all commands overview" on page 343
- "Specifying the rah and db2_all commands" on page 345

Related reference:

- "rah and db2_all command descriptions" on page 344

rah and db2_all commands overview

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"
```

Related concepts:

- "Issuing commands in a partitioned database environment" on page 343
- "Specifying the rah and db2_all commands" on page 345

Related reference:

- "rah and db2_all command descriptions" on page 344

rah and db2_all 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.

Related concepts:

Specifying the rah and db2_all commands

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" &
```

Related concepts:

- "Running commands in parallel on UNIX-based platforms" on page 346
- "Additional rah information (Solaris and AIX only)" on page 348

Related tasks:

- "Setting the default environment profile for rah on Windows NT" on page 354

Related reference:

- "rah and db2_all command descriptions" on page 344
- "rah command prefix sequences" on page 348
- "Controlling the rah command" on page 352

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.

Related concepts:

- "Additional rah information (Solaris and AIX only)" on page 348

Related tasks:

- "Monitoring rah processes on UNIX-based platforms" on page 347

Related reference:

- "rah command prefix sequences" on page 348
- "Determining problems with rah on UNIX-based platforms" on page 354

Monitoring rah processes on UNIX-based platforms

Procedure:

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.

Related concepts:

- "Running commands in parallel on UNIX-based platforms" on page 346
- "Additional rah information (Solaris and AIX only)" on page 348

Additional rah 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.)

Related concepts:

- "Running commands in parallel on UNIX-based platforms" on page 346

Related tasks:

- "Monitoring rah processes on UNIX-based platforms" on page 347

rah command 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) 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.
2. When " is specified, duplicates are not eliminated from the list of machines.

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.

Related concepts:

- "Specifying the rah and db2_all commands" on page 345
- "Running commands in parallel on UNIX-based platforms" on page 346

Related reference:

- "rah and db2_all command descriptions" on page 344

Specifying the list of machines in a partitioned environment

Procedure:

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.

Related tasks:

- "Eliminating duplicate entries from a list of machines in a partitioned environment" on page 351

Eliminating duplicate entries from a list of machines in a partitioned environment

Procedure:

If you are running DB2 Universal Database Enterprise Server 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.

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.

Related tasks:

- "Specifying the list of machines in a partitioned environment" on page 350

Related reference:
- "RESTART DATABASE Command" in the *Command Reference*
- "rah command prefix sequences" on page 348

Controlling the rah command

You can use the following environment variables to control the **rah** command.

Table 51.

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
$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

Table 51. (continued)

Name	Meaning	Default
$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.

Related reference:

- "Using $RAHDOTFILES on UNIX-based platforms" on page 353

Using $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
```

Related reference:

- "Controlling the rah command" on page 352

Setting the default environment profile for rah on Windows NT

Procedure:

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**.

Related concepts:

- "Specifying the rah and db2_all commands" on page 345

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.

Related reference:

- "Controlling the rah command" on page 352

Appendix E. Using Windows Management Instrumentation (WMI) support

Introduction to Windows Management Instrumentation (WMI)

There is an industry initiative that establishes management infrastructure standards and provides a way to combine information from various hardware and software management systems. This initiative is called Web-Based Enterprise Management (WBEM). WBEM is based on the Common Information Model (CIM) schema, which is an industry standard driven by the Desktop Management Task Force (DMTF).

Microsoft® Windows® Management Instrumentation (WMI) is an implementation of the WBEM initiative for supported Windows platforms. WMI is useful in a Windows enterprise network where it reduces the maintenance and cost of managing enterprise network components. WMI provides:

- A consistent model of Windows operation, configuration, and status.
- A COM API to allow access to management information.
- The ability to operate with other Windows management services.
- A flexible and extensible architecture allowing vendors a means of writing other WMI providers to support new devices, applications, and other enhancements.
- The WMI Query Language (WQL) to create detailed queries of the information.
- An API for management application developers to write Visual Basic or Windows Scripting Host (WSH) scripts.

The WMI architecture has two parts:

1. A management infrastructure that includes the CIM Object Manager (CIMOM) and a central storage area for management data called the CIMOM object repository. CIMOM allows applications to have a uniform way to access management data.
2. WMI providers. WMI providers are the intermediaries between CIMOM and managed objects. Using WMI APIs, WMI providers supply CIMOM with data from managed objects, handle requests on behalf of management applications, and generate event notifications.

Windows Management Instrumentation (WMI) providers are standard COM or DCOM servers that function as mediators between managed objects and the CIM Object Manager (CIMOM). If the CIMOM receives a request from a management application for data that is not available from the CIMOM object repository, or for events, the CIMOM forwards the request to the WMI providers. WMI providers supply data, and event notifications, for managed objects that are specific to their particular domain.

Related concepts:

- "DB2 Universal Database integration with Windows Management Instrumentation" on page 358

DB2 Universal Database integration with Windows Management Instrumentation

The snapshot monitors can be accessed by Windows® Management Instrumentation (WMI) by means of the DB2® performance counters and using the built-in PerfMon provider.

The DB2 profile registry variables can be accessed by WMI by using the built-in Registry provider.

The WMI Software Development Kit (WMI SDK) includes several built-in providers:

- PerfMon provider
- Registry event provider
- Registry provider
- Windows NT® event log provider
- Win32 provider
- WDM provider

The DB2 errors that are in the Event Logs can be accessed by WMI by using the built-in Windows NT Event Log provider.

DB2 Universal Database™ (UDB) has a DB2 WMI Administration provider, and sample WMI script files, to access the following managed objects:

1. Instances of the database server including those instances that are partitioned. The following operations can be done:
   - Enumerate instances
   - Configure database manager parameters
   - Start/stop/query the status of the DB2 server service
   - Setup or establish communication
2. Databases. The following operations can be done:
   - Enumerate databases
   - Configure database parameters
   - Create/drop databases
   - Backup/restore/roll forward databases

You will need to register the DB2 WMI provider with the system before running WMI applications. Registration is done by entering the following commands:

- mofcomp %DB2PATH%\bin\db2wmi.mof

  This command loads the definition of the DB2 WMI schema into the system.
- regsvr %DB2PATH%\bin\db2wmi.dll

  This command registers the DB2 WMI provider COM DLL with Windows.

In both commands, %DB2PATH% is the path where DB2 is installed. Also, db2wmi.mof is the .MOF file that contains the DB2 WMI schema definition.

There are several benefits to integrating with the WMI infrastructure:

1. You are able to easily write scripts to manage DB2 servers in a Windows-based environment using the WMI provided tool. Sample Visual Basic (VBS) scripts are provided to carry out simple tasks such as listing instances, creating and

dropping databases, and updating configuration parameters. The sample scripts are included in the DB2 Application Development for Windows product.

2. You can create powerful management applications that perform many tasks using WMI. The tasks could include:
   - Displaying system information
   - Monitoring DB2 performance
   - Monitoring DB2 system resource consumption

   By monitoring both system events and DB2 events through this type of management application, you can manage the database better.
3. You can use existing COM and Visual Basic programming knowledge and skills. By providing a COM or Visual Basic interface, your programmers can save time when developing enterprise management applications.

Related concepts:

- "Introduction to Windows Management Instrumentation (WMI)" on page 357

Appendix F. Using Windows NT security

DB2 for Windows NT and Windows NT security introduction

A Windows® NT domain is an arrangement of client and server computers referenced by a specific and unique name; and, that share a single user accounts database called the Security Access Manager (SAM). One of the computers in the domain is the domain controller. The domain controller manages all aspects of user-domain interactions. The domain controller uses the information in the domain user accounts database to authenticate users logging onto domain accounts. For each domain, one domain controller is the primary domain controller (PDC). Within the domain, there may also be backup domain controllers (BDC) which authenticate user accounts when there is no primary domain controller or the primary domain controller is not available. Backup domain controllers hold a copy of the SAM database which is regularly synchronized against the master copy on the PDC.

User accounts, user IDs, and passwords only need to be defined at the primary domain controller to be able to access domain resources.

During the setup procedure when a Windows NT® server is installed, you may select to create:

- A primary domain controller in a new domain
- A backup domain controller in a known domain
- A stand-alone server in a known domain.

Selecting "controller" in a new domain makes that server the primary domain controller.

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:

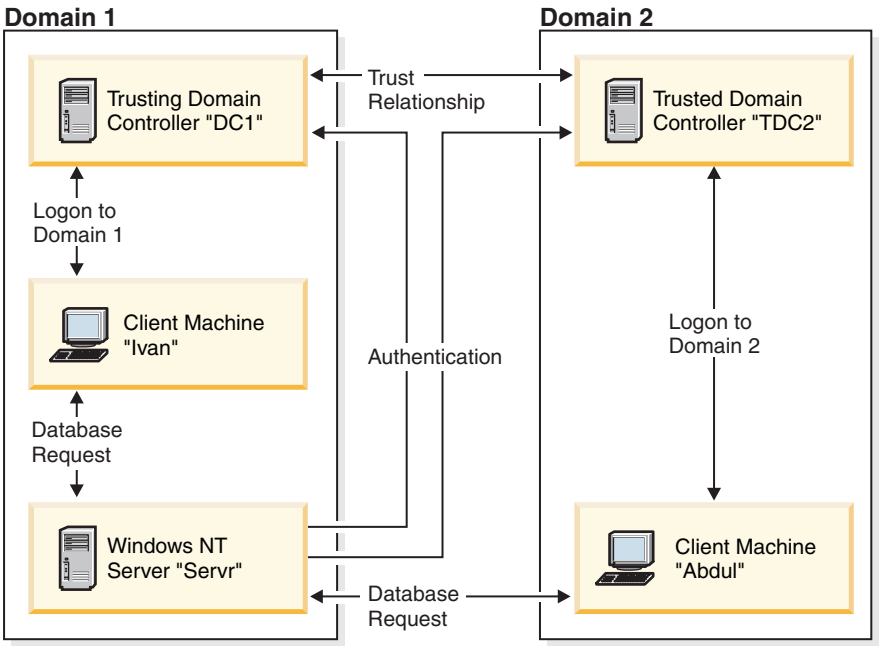


Figure 6. Authentication Using Windows NT Domains

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.

Related concepts:

- "Groups and user authentication on Windows NT" on page 365

Related tasks:

- "Using a backup domain controller with DB2 UDB" on page 364
- "Installing DB2 on a backup domain controller" on page 367
- "DB2 for Windows NT authentication with groups and domain security" on page 368

A DB2 for Windows NT 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.

Related concepts:

• "DB2 for Windows NT and Windows NT security introduction" on page 361

A DB2 for Windows NT 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
   db2start
   ```
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 as part of the Windows installation. DB2 is then installed and "registered" as a Windows NT service however, it is not started automatically. To start the DB2 Security Service, enter the `NET START DB2NTSECSERVER` command.

Related concepts:

• "DB2 for Windows NT and Windows NT security introduction" on page 361

A DB2 for Windows NT scenario with client authentication and a Windows 9x 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
   db2start
   ```
2. Ivan, at a Windows 9x 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 9x 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 9x client cannot validate a given username and password, client authentication under Windows 9x is inherently insecure. If the Windows 9x machine has access to a Windows NT security provider, however, some measure of security can be imposed by configuring the Windows 9x system for validated pass-through logon. For details on how to configure your Windows 9x system in this way, refer to the Microsoft documentation for Windows 9x.

Related concepts:

- "DB2 for Windows NT and Windows NT security introduction" on page 361
- "Support for global groups (on Windows)" on page 364

Support for global groups (on Windows)

DB2® Universal Database (DB2 UDB) also supports global groups. In order to use global groups, you must include global groups inside a local group. When DB2 UDB 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).

Global groups are used in two possible situations:

- Included inside a local group. Permission must be granted to this local group.
- Included on a domain controller. Permission must be granted to the global group.

Related concepts:

- "Groups and user authentication on Windows NT" on page 365

Using a backup domain controller with DB2 UDB

Procedure:

If the server you use for DB2 Universal Database™ (DB2 UDB) also acts as a backup domain controller, you can improve DB2 UDB performance and reduce network traffic if you configure DB2 UDB to use the backup domain controller.

You specify the backup domain controller to DB2 UDB by setting the DB2DMNBCKCTLR registry variable.

If you know the name of the domain for which DB2 UDB server is the backup domain controller, use:

```
db2dmnbckctlr=<domain_name>
```

where `domain_name` must be in upper case.

To have DB2 UDB determine the domain for which the local machine is a backup domain controller, use:

```
DB2DMNBCKCTLR=?
```

Note: DB2 UDB 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.

Related tasks:

- "Installing DB2 on a backup domain controller" on page 367

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:

- "DB2 for Windows NT user name and group name restrictions"
- "DB2 for Windows NT security service" on page 367
- "Installing DB2 on a backup domain controller" on page 367
- "DB2 for Windows NT authentication with groups and domain security" on page 368

DB2 for Windows NT user name and group name restrictions

The following are the limitations in this environment:

- User names and group names are limited to 30 characters within DB2 Universal Database™ (DB2 UDB).
- 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 UDB. 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.)
- A user can not belong to more than 64 groups.
- DB2 UDB supports a single namespace. That is, when running in a trusted domains environment, you should not have a user account of the same name that exists in multiple domains, or that exists in the local SAM of the server machine and in another domain.

Related concepts:

- "Groups and user authentication on Windows NT" on page 365
- "Trust relationships between domains on Windows NT" on page 366

Groups and user authentication on Windows NT

Users are defined on Windows® NT by creating user accounts using the Windows NT® administration tool called the "User Manager".

An account containing other accounts, also called members, is a group. Groups give Windows NT administrators the ability to grant rights and permissions to the users within the group at the same time, without having to maintain each user individually. Groups, like user accounts, are defined and maintained in the Security Access Manager (SAM) database.

There are two types of groups:

- Local groups. A local group can include user accounts created in the local accounts database. If the local group is on a machine that is part of a domain, the local group can also contain domain accounts and groups from the Windows NT domain. If the local group is created on a workstation, it is specific to that workstation.
- Global groups. A global group exists only on a domain controller and contains user accounts from the domain's SAM database. That is, a global group can only contain user accounts from the domain on which it is created; it cannot contain any other groups as members. A global group can be used in servers and workstations of its own domain, and in trusting domains.

Related concepts:

- "Trust relationships between domains on Windows NT" on page 366
- "Support for global groups (on Windows)" on page 364

Related tasks:

- "DB2 for Windows NT authentication with groups and domain security" on page 368

Related reference:

- "DB2 for Windows NT user name and group name restrictions" on page 365

Trust relationships between domains on Windows NT

Trust relationships are an administration and communication link between two domains. A trust relationship between two domains enables user accounts and global groups to be used in a domain other than the domain where the accounts are defined. Account information is shared to validate the rights and permissions of user accounts and global groups residing in the trusted domain without being authenticated. Trust relationships simplify user administration by combining two or more domains into an single administrative unit.

There are two domains in a trust relationship:

- The trusting domain. This domain trusts another domain to authenticate users for them.
- The trusted domain. This domain authenticates users on behalf of (in trust for) another domain.

Trust relationships are not transitive. This means that explicit trust relationships need to be established in each direction between domains. For example, the trusting domain may not necessarily be a trusted domain.

Related concepts:

- "Groups and user authentication on Windows NT" on page 365
- "Support for global groups (on Windows)" on page 364

Related reference:

- "DB2 for Windows NT user name and group name restrictions" on page 365

DB2 for Windows NT security service

In DB2® Universal Database (DB2 UDB) 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.

Related concepts:

- "DB2 for Windows NT and Windows NT security introduction" on page 361

Installing DB2 on a backup domain controller

Procedure:

In a Windows NT 4.0 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.)

Related tasks:

- "Using a backup domain controller with DB2 UDB" on page 364

DB2 for Windows NT authentication with groups and domain security

Procedure:

DB2 Universal Database™ (DB2 UDB) allows you to specify either a local group or a global group when granting privileges or defining authority levels. A user is determined to be a member of a group if the user's account is defined explicitly in the local or global group, or implicitly by being a member of a global group defined to be a member of a local group.

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 UDB provides an override that forces group enumeration to occur on the local Windows NT server where DB2 UDB 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 <instance name> DB2_GRP_LOOKUP=local
    ```

    After issuing this command, you must stop and start the DB2 UDB instance for the change to take effect. Then create local groups and include domain accounts or global groups in the local group.

  To view all DB2 profile registry variables that are set, type

  ```
  db2set -all
  ```

If the DB2_GRP_LOOKUP profile registry variable is set to local, then DB2 UDB 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 UDB 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 UDB attempts to find the user name on it domain, and then on trusted domains.

If DB2 UDB is running on a machine that is a primary or backup domain controller in the resource domain, it is able to locate any domain controller in any trusted domain. This occurs because the names of the domains of backup domain controllers in trusted domains are only known if you are a domain controller.

If DB2 UDB is not running on a domain controller, then you should issue:

```
db2set -g DB2_GRP_LOOKUP=DOMAIN
```

This command tells DB2 UDB to use a domain controller in its own domain to find the name of a domain controller in the accounts domain. That is, when DB2 UDB finds out that a particular user account is defined in domain x, rather than attempting to locate a domain controller for domain x, it sends that request to a domain controller in its own domain. The name of the domain controller in the account domain will be found and returned to the machine DB2 UDB is running on. There are two advantages to this method:

1. A backup domain controller is found when the primary domain controller is unavailable.
2. A backup domain controller is found that is close when the primary domain controller is geographically remote.

Related concepts:

• "Groups and user authentication on Windows NT" on page 365

Authentication using an ordered domain list

User IDs may be defined more than once in a trusted domain forest. A trusted domain forest is a collection of domains that are interrelated through a network. It is possible for a user on one domain to have the same user ID as that for another user on a different domain. This may cause difficulties when attempting to do any of the following:

- Authenticating multiple users having the same user ID but on different domains.
- Group lookup for the purposes of granting and revoking privileges based on groups.
- Validation of passwords.
- Control of network traffic.

Procedure:

To prevent the difficulties arising from the possibility of multiple users with the same user ID across across a domain forest, you should use an ordered domain list as defined using the **db2set** and the registry variable DB2DOMAINLIST. When setting the order, the domains to be included in the list are separated by a comma. You must make a conscious decision regarding the order that the domains are searched when authenticating users.

Those user IDs that are present on domains further down the domain list will have to be renamed by you if they are to be authenticated for access.

Control of access can be done through the domain list. For example, if the domain of a user is not in the list, the user will not be allowed to connect.

Note: The DB2DOMAINLIST registry variable is effective only when CLIENT authentication is set in the database manager configuration and is needed if a single signon from a Windows NT desktop is required in a Windows NT domain environment.

Related concepts:

- "DB2 for Windows NT and Windows NT security introduction" on page 361

DB2 for Windows NT support of domain security

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 52. 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.	

Related concepts:

- "Groups and user authentication on Windows NT" on page 365

Related tasks:

- "DB2 for Windows NT authentication with groups and domain security" on page 368

Appendix G. Using the Windows Performance Monitor

Windows performance monitor introduction

When working with DB2® Universal Database (DB2 UDB) for Windows®, there are tools that can be used to monitor performance:

- **DB2 Performance Expert**

  DB2 Performance Expert for Multiplatforms, Version 1.1 consolidates, reports, analyzes and recommends self-managing and resource tuning changes based on DB2 UDB performance-related information.
- **DB2 UDB Health Center**

  The functions of the Health Center provide you with different methods to work with performance-related information. These functions somewhat replace the performance monitor capability of the Control Center.
- **Windows Performance Monitor**

  The Windows 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 also provides performance information data on all aspects of machine operation including:
  - CPU usage
  - Memory utilization
  - Disk activity
  - Network activity

Related tasks:

- "Registering DB2 with the Windows performance monitor" on page 371
- "Enabling remote access to DB2 performance information" on page 372
- "Displaying DB2 UDB and DB2 Connect performance values" on page 373
- "Accessing remote DB2 UDB performance information" on page 374
- "Resetting DB2 performance values" on page 374

Related reference:

- "Windows performance objects" on page 373

Registering DB2 with the Windows performance monitor

Procedure:

The setup program automatically registers DB2 with the Windows Performance Monitor for you.

To make DB2 Universal Database™ (DB2 UDB) and DB2 Connect performance information accessible to the Windows Performance Monitor, you must register the DLL for the DB2 for Windows Performance Counters. This also enables any other Windows application using the Win32 performance APIs to get performance data.

To install and register the DB2 for Windows Performance Counters DLL (DB2Perf.DLL) with the Windows 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.

Related reference:

- "db2perfi - Performance Counters Registration Utility Command" in the *Command Reference*

Enabling remote access to DB2 performance information

Procedure:

If your DB2 for Windows workstation is networked to other Windows machines, you can use the feature described in this section.

In order to see Windows performance objects from another DB2 for Windows machine, you must register an administrator username and password with DB2 Universal Database™ (DB2 UDB). (The default Windows Performance Monitor username, **SYSTEM**, is a DB2 UDB reserved word and cannot be used.) To register the name, type:

```
db2perfr -r username password
```

Note: The `username` used must conform to the DB2 UDB 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 UDB, 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 UDB does not match, local sessions of the Performance Monitor will not show DB2 UDB performance information.
2. The username and password combination must be maintained to match the username and password values stored in the Windows Security database. If the username or password is changed in the Windows Security database, the username and password combination used for remote performance monitoring must be reset.
3. To deregister, type:

   ```
   db2perfr -u <username> <password>
   ```

Related concepts:

- "General naming rules" on page 291

Related reference:

- "db2perfr - Performance Monitor Registration Tool Command" in the *Command Reference*

Displaying DB2 UDB and DB2 Connect performance values

Procedure:

To display DB2 Universal Database™ (DB2 UDB) 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:".

Related concepts:

- "Windows performance monitor introduction" on page 371

Related reference:

- "Windows performance objects" on page 373

Windows performance objects

Windows provides the following performance objects:

- **DB2 Database Manager**

  This object provides general information for a single Windows instance. The DB2 Universal Database™ (DB2 UDB) instance being monitored appears as the object instance.

  For practical and performance reasons, you can only get performance information from one DB2 UDB instance at a time. The DB2 UDB instance that the Performance Monitor shows is governed by the `db2instance` registry variable in the Performance Monitor process. If you have multiple DB2 UDB 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 UDB 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 UDB 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 UDB application. Information is available for each currently active DB2 UDB 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 Performance Monitor depends on what is installed on your Windows 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 UDB 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 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.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Accessing remote DB2 UDB performance information

Procedure:

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.

Related concepts:

- "Windows performance monitor introduction" on page 371

Resetting DB2 performance values

Procedure:

When an application calls the DB2 monitor APIs, the information returned is normally the cumulative values since the DB2 Universal Database™ (DB2 UDB) 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 UDB 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 UDB databases. The next example resets values for specific DB2 UDB 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 UDB 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 UDB performance information when the **db2perfc** command is executed.

Note: There is a DB2 UDB API, sqlmrset, that allows an application to reset the values it sees locally, not globally, for particular databases.

Related reference:

- "db2ResetMonitor - Reset Monitor" in the *Administrative API Reference*
- "db2perfc - Reset Database Performance Values Command" in the *Command Reference*

Appendix H. Using Windows database partition servers

When working to change the characteristics of your configuration in a Windows environment, the tasks involved are carried out using specific utilities.

The utilities presented here are:

- "Listing database partition servers in an instance"
- "Adding a database partition server to an instance (Windows)"
- "Changing the database partition (Windows)" on page 379
- "Dropping a database partition from an instance (Windows)" on page 380

Listing database partition servers in an instance

Procedure:

On Windows, 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.

Related tasks:

- "Adding a database partition server to an instance (Windows)" on page 377
- "Changing the database partition (Windows)" on page 379
- "Dropping a database partition from an instance (Windows)" on page 380

Adding a database partition server to an instance (Windows)

Procedure:

On Windows, 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 `%SystemRoot%\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 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
```

Related reference:

- "db2start - Start DB2 Command" in the *Command Reference*
- "db2icrt - Create Instance Command" in the *Command Reference*
- "db2ncrt - Add Database Partition Server to an Instance Command" in the *Command Reference*

Changing the database partition (Windows)

Procedure:

On Windows, 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 Universal Database™ (DB2 UDB) 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
```

DB2 UDBprovides 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. The db2set command would appear as:

```
db2set DB2REMOTEPREG=<remote workstation>
```

where <remote workstation> is the remote workstation name.

Note: Care should be taken in setting this option since all DB2 UDB 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.

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related reference:

- "db2nchg - Change Database Partition Server Configuration Command" in the *Command Reference*

Dropping a database partition from an instance (Windows)

Procedure:

On Windows, 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).

Related concepts:

- "DB2 registry and environment variables" in the *Administration Guide: Performance*

Related reference:

- "db2stop - Stop DB2 Command" in the *Command Reference*
- "db2idrop - Remove Instance Command" in the *Command Reference*
- "db2ndrop - Drop Database Partition Server from an Instance Command" in the *Command Reference*

Appendix I. Configuring multiple logical nodes

When to use multiple logical nodes

Typically, you configure DB2® Universal Database (DB2 UDB) Enterprise Server 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.

Related tasks:

- "Configuring multiple logical nodes" on page 383

Related reference:

- "db2start - Start DB2 Command" in the *Command Reference*

Configuring multiple logical nodes

Procedure:

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
```

Use the fully-qualified name for the hostname. The `/etc/hosts` file also should use the fully-qualified name. If the fully-qualified name is not used in the db2nodes.cfg file and in the `/etc/hosts` file, you may receive error message SQL30082N RC=3.

You must ensure that you define enough ports in the `services` file of the `etc` directory for FCM communications.

Related concepts:

- "When to use multiple logical nodes" on page 383

Related tasks:

- "Changing node and database configuration files" on page 149
- "Creating a node configuration file" on page 33

Related reference:

- "db2start - Start DB2 Command" in the *Command Reference*
- "db2ncrt - Add Database Partition Server to an Instance Command" in the *Command Reference*

Appendix J. Extending the Control Center

Introducing the plug-in architecture for the Control Center

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, add objects to the Control Center tree, 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.

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, for most users, the plug-in ZIP file, will be small and the impact should be minimal.

Related concepts:

- "Compiling and running the example plugins" on page 386
- "Writing plugins as Control Center extensions" on page 387
- "Guidelines for Control Center plugin developers" on page 385

Guidelines for Control Center plugin developers

Since multiple plugins can be contained in the db2plug.zip file, plugin developers should follow these guidelines when creating a plugin for the Control Center:

- Use Java™ packages to ensure your plugin classes have unique names. Follow the Java package naming convention. Prefix your package names with the inverted name of your Internet domain (for example, com.companyname). All package names, or at least their unique prefixes, should be lowercase letters.
- db2plug.zip should be installed in the tools directory under the `sqllib` directory. Before V8, the db2plug.zip needed to be installed in the `cc` directory under the `sqllib` directory.
- When you create a plugin for the Control Center and a db2plug.zip file already exists, you should add your plugin classes to the existing db2plug.zip. You should not overwrite the existing db2plug.zip file with your own db2plug.zip file. To add your plugin to an existing db2plug.zip, the following zip command should be used:

  ```
  zip -r0 db2plug.zip com\companyname\myplugin\*.class
  ```

  where your plugin package name is com.companyname.myplugin
- All the classes in the db2plug.zip get loaded when the Control Center is started. The db2plug.zip file should contain all your CCExtension class files and classes which extend or implement classes in the com.ibm.db2.tools.cc.navigator package. Other classes not used directly by these classes do not need to be included in the db2plug.zip. They can be stored in a separate jar file to minimize performance impacts when the Control Center is started. This is a good idea if there are a large number of extra classes. You should put your jar file in the

tools directory under the sqllib directory. Your jar file will automatically be included in the *classpath* when the **db2cc** command is used to start the Control Center.

- Plugin classes which implement CCObject should provide a no-argument default constructor to allow calls to Class.newInstance() by the Control Center.
- Where possible avoid using inner classes. In general, Plugin classes which implement CCTreeObject to create new plugin objects in the Control Center should not be declared as inner classes. This will prevent the Control Center from instantiating these classes.
- Test that your plugin is loaded correctly by using **db2cc -tf filename**. This will put Control Center trace information in the specified filename. If you do not provide a full pathname, the trace file will be written to the tools directory in sqllib. Plugin related trace statements will contain the word "Plugin". You can see if your classes were loaded by looking for lines containing the text "PluginLoader".

Related concepts:

- "Compiling and running the example plugins" on page 386
- "Writing plugins as Control Center extensions" on page 387

Related reference:

- "db2cc - Start Control Center Command" in the *Command Reference*

Compiling and running the example plugins

The Control Center Plugin function is demonstrated in the sections that follow and their corresponding plugin sample programs: Example1.java, Example2.java, Example3.java, Example3Folder.java, and Example3Child.java. These example java files are installed with the DB2® Application Development client. On Windows® platforms, these sample programs are in DRIVE:\sqllib\samples\java\plugin where DRIVE: represents the drive on which DB2 is installed. On UNIX® platforms, these samples are in /u/db2inst1/sqllib/samples/java/plugin where /u/db2inst1 represents the directory in which DB2 is installed.

Note: The plugin sample programs may contain updates which are not yet reflected here. The example code and java documentation should be considered the most current information when there are differences with what is shown here.

To run the example plugins, you must ZIP the extension class files according to the rules of a Java™ archive file. The ZIP file (db2plug.zip) must be in the *classpath*. On Windows operating systems, put db2plug.zip in the DRIVE:\sqllib\tools directory where DRIVE: represents the drive on which DB2 is installed. On UNIX platforms, put db2plug.zip in the /u/db2inst1/sqllib/tools directory where /u/db2inst1 represents the directory on which DB2 is installed.

Note: The **db2cc** command sets the *classpath* to point to db2plug.zip in the tools directory.

The examples (except Example3, Example3Folder, and Example3Child which go together) should not be zipped into the same db2plug.zip since they may conflict with one another.

To compile any of these example java files, the following must be included in your *classpath*:

- On Windows platforms use:
  - `DRIVE: \sqllib\java\Common.jar`
  - `DRIVE: \sqllib\tools\db2navplug.jar`

  where `DRIVE` represents the drive on which DB2 is installed.
- On UNIX platforms use:
  - `/u/db2inst1/sqllib/java/Common.jar`
  - `/u/db2inst1/sqllib/tools/db2navplug.jar`

  where `/u/db2inst1` represents the directory in which DB2 is installed.

Create the db2plug.zip to include all the classes generated from compiling the example java file. The file should not be compressed. For example, issue the following:

```
zip -r0 db2plug.zip *.class
```

This command places all the class files into the db2plug.zip file and preserves the relative path information.

Related concepts:

- "Writing plugins as Control Center extensions" on page 387
- "Guidelines for Control Center plugin developers" on page 385

Related reference:

- "db2cc - Start Control Center Command" in the *Command Reference*

Writing plugins as Control Center extensions

The first step to writing a plugin is to define a class that implements the CCExtension interface. This class will contain the list of plugin classes to be loaded by the Control Center. If you want to add menu items to the standard Control Center objects such as Databases and Tables, or want to create your own objects for display in the tree, you create classes that implement the CCObject interface and return and array of these CCObjects in the getObjects method. If you want to add a toolbar button, you implement CCToolbarAction and return an array of CCToolbarActions in the getToolbarActions method.

Each of these interfaces is documented in:

- On Windows® platforms, in `DRIVE:\sqllib\samples\java\plugin\doc` where `DRIVE:` represents the drive on which DB2® is installed.
- On UNIX® platforms, in `/u/db2inst1/sqllib/samples/java/plugin/doc` where `/u/db2inst1` represents the directory in which DB2 is installed.

Related tasks:

- "Creating a plugin that adds a toolbar button" on page 388
- "Creating a basic menu action" on page 389
- "Positioning the menu item" on page 391
- "Creating a basic menu action separator" on page 391
- "Creating sub menus" on page 392
- "Adding a menu item only to an object with a particular name" on page 393

Plug-in task descriptions

The following plug-in tasks are discussed:

1. Creating a plug-in that adds a toolbar button
2. Creating a plug-in that adds new menu items to the Database object
3. Creating a plug-in that adds plug-in objects under Database in the tree
4. Disabling configuration features with isConfigurable()
5. Disabling the ability to alter objects using isEditable()
6. Disabling the default buttons in configuration dialogs using hasConfigurationDefaults()

Creating a plugin that adds a toolbar button

Procedure:

For this example, we will just be adding a toolbar button, so getObjects should return a null array, as follows:

```
import  com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import  javax.swing.*;

public class Example1 implements CCExtension {

  public CCObject[] getObjects () {
    return null;
  }

}
```

Notice that the com.ibm.db2.tools.cc.navigator package is imported. This class will implement the CCToolbarAction interface which requires implementing three methods: getHoverHelpText, getIcon, and actionPerformed. The Control Center uses getHoverHelpText to display the small box of text that appears when a user leaves a mouse hovering over your toolbar button. You specify the icon for your button using getIcon. The Control Center calls actionPerformed when a user clicks on your button. Here is an example that adds a button named X that writes a message to the console when you click it. It uses the Refresh icon from the Control Center's image repository class.

```
class Example1ToolbarAction implements CCToolbarAction {

  public String getHoverHelpText() { return "X"; }

  public ImageIcon getIcon() {
```

```
   return CommonImageRepository.getCommonIcon(CommonImageRepository.WC_NV_
REFRESH);
  }

  public void actionPerformed(ActionEvent e) {
   System.out.println("I've been clicked");
  }

}
```

The final step is to implement the getToolbarActions method in Example1 to return an instance of your new class, as follows:

```
  public CCToolbarAction[] getToolbarActions () {
   return new CCToolbarAction[] { new Example1ToolbarAction() };
  }
```

Related concepts:

- "Compiling and running the example plugins" on page 386

Creating a plug-in that adds new menu items to the Database object

The following procedure outlines how to create a plug-in that adds new menu items to the Database object:

1. Creating the basic menu action
2. Positioning the menu item
3. Creating a basic menu action separator
4. Creating submenus
5. Adding a menu item only to an object with a particular name

Creating a basic menu action

Procedure:

In this slightly more advanced topic, we will add new commands to the popup menu of the Database object.

As in Example 1, the first step is to write a class that extends CCExtension.

```
import  com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import  javax.swing.*;

public class Example2 implements CCExtension {

  public CCToolbarAction[] getToolbarActions () {
    return null;
  }

}
```

The second step is to create a CCObject for the Database object in the tree, as follows:

```
  class CCDatabase implements CCObject {

    public String getName () { return null; }
    public boolean isEditable () { return true; }
```

```
  public boolean isConfigurable () { return true; }

  public int getType () { return UDB_DATABASE; }
}
```

Because we are not using any features other than the ability to add menu items to Control Center built-in objects (for example, the Database object in this example), we implement most functions as returning null or true. To specify that we want this object to represent the DB2 UDB Database object, we specify its type as UDB_DATABASE, a constant in CCObject. The class is named CCDatabase in this example, but you should make your class names as unique as possible, since there may be other vendor's plugins in the same zip file as your plugin. Java packages should be used to help ensure unique class names.

The getObjects method of your CCExtension should return an array containing an instance of CCDatabase as follows:

```
public CCObject[] getObjects () {
  return new CCObject[] { new CCDatabase() };
}
```

You can create multiple CCObject subclasses whose type is UDB_DATABASE, but if the values returned from their isEditable or isConfigurable methods conflict, the objects that return false override those that return true.

The only remaining method to implement is getMenuActions. This returns an array of CCMenuActions, so first we will write a class that implements this interface.

There are two methods to implement: getMenuText and actionPerformed. The text displayed in the menu is obtained using getMenuText. When a user clicks your menu item, the event that is triggered results in a call to actionPerformed.

The following example class displays a menu item called ″Example2a Action″ when a single database object is selected. When the user clicks this menu item, the message ″Example2a menu item actionPerformed″ is written to the console.

```
class Example2AAction implements CCMenuAction {

  public String getMenuText () { return "Example2a Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2a menu item actionPerformed");
  }

}
```

Finally, attach this menu item to your UDB Database CCObject by adding the following to your CCObject.

```
  public CCMenuAction[] getMenuActions () {
     return new CCMenuAction[] { new Example2AAction() };
  }
```

Related concepts:

- "Compiling and running the example plugins" on page 386

Related tasks:

- "Positioning the menu item" on page 391
- "Creating a basic menu action separator" on page 391

- "Creating sub menus" on page 392
- "Adding a menu item only to an object with a particular name" on page 393

Positioning the menu item

Procedure:

When creating the basic menu item, we did not specify the position of the menu item within the menu. The default behavior when adding plugin menu items to a menu is to add them on the end, but before any Refresh and Filter menu items.

You can override this behavior to specify any position number from zero up to the number of items in the menu, not counting the Refresh and Filter menu items. Change your CCMenuAction subclass to implement Positionable and then implement the getPosition method, as follows:

```
class Example2BAction implements CCMenuAction, Positionable {

  public String getMenuText () { return "Example2B Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2B menu item actionPerformed");
  }

  public int getPosition() {
     return 0;
  }
}
```

Specifying a position number of zero places your menu item as the first in the list and specifying a position number equal to the number of items in the menu not counting your plugin menu item puts it at the bottom, but before any Refresh and Filter menu items. You can also return a value of Positionable.POSITION_BOTTOM to get the default behavior, that is, have your menu item placed at the bottom before any Refresh and Filter menu items. If there is more than one CCObject of type UDB_DATABASE with menu items positioned at POSITION_BOTTOM, the menu items are ordered based on the order in which the CCObjects of type UDB_DATABASE are returned from the getObjects method in the CCExtension.

Change CCDatabase to add Example2BAction to the menu as follows:

```
public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction() };
}
```

Related tasks:

- "Creating a basic menu action" on page 389
- "Creating a basic menu action separator" on page 391
- "Creating sub menus" on page 392
- "Adding a menu item only to an object with a particular name" on page 393

Creating a basic menu action separator

Procedure:

To add a separator, create a CCMenuAction that implements the Separator interface. All other methods (except getPosition if you implement Positionable) will be ignored.

```
class Example2CSeparator implements CCMenuAction, Separator, Positionable {

  public String getMenuText () { return null; }

  public void actionPerformed (ActionEvent e) {}

  public int getPosition() {
     return 1;
  }
}

public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction(),
                              new Example2CSeparator() };
}
```

Related tasks:

- "Creating a basic menu action" on page 389
- "Positioning the menu item" on page 391
- "Creating sub menus" on page 392
- "Adding a menu item only to an object with a particular name" on page 393

Creating sub menus

Procedure:

A sub-menu is just an array of CCMenuActions. To have a menu item contain sub-menus, it must implement the SubMenuParent interface. Then create an implementation of CCMenuAction for each submenu item and return them in an array from the getSubMenuActions method of the SubMenuParent interface. Adding menu items to non-plugin submenus is not supported. Also, note that SubMenuParents do not receive ActionEvents from the Control Center. Here is an example:

```
class Example2DAction implements CCMenuAction, SubMenuParent {

  public String getMenuText () { return "Example2D Action"; }

  public void actionPerformed (ActionEvent e) {}

  public CCMenuAction[] getSubMenuActions() {
     return new CCMenuAction[] { new Example2DSubMenuAction() };
  }

}

class Example2DSubMenuAction implements CCMenuAction {

  public String getMenuText () { return "Example2D Sub-Menu Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2D sub-menu menu item actionPerformed");
  }

}
```

Once again, add this new menu item to CCDatabase.

```
public CCMenuAction[] getMenuActions () }
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction(),
                              new Example2CSeparator(),
                              new Example2DAction() };
}
```

Related tasks:

- "Creating a basic menu action" on page 389
- "Positioning the menu item" on page 391
- "Creating a basic menu action separator" on page 391
- "Adding a menu item only to an object with a particular name" on page 393

Adding a menu item only to an object with a particular name

Procedure:

Currently, any database you display in the Control Center will show the plugin menu items you've written. You can restrict these menu items to a database of a particular name by returning that name in the getName method of CCDatabase. This must be a fully qualified name. In this case, since we are referring to a database, we must include the system, instance and database names in what we return in the getName method. These names are separated by " – ". Here is an example for a system named MYSYSTEM, an instance named DB2, and a database named SAMPLE.

```
class CCDatabase implements CCObject {
  ...
  public String getName () { return "MYSYSTEM – DB2 – SAMPLE"; }
  ...
}
```

Related tasks:

- "Creating a basic menu action" on page 389
- "Positioning the menu item" on page 391
- "Creating a basic menu action separator" on page 391
- "Creating sub menus" on page 392

Creating a plug-in that adds plug-in objects under Database in the tree

The following procedure outlines how to create a plug-in that adds plug-in objects under Database in the tree:

1. Adding the folder to hold multiple objects in the tree
2. Adding an example object under the folder
3. Setting attributes for a plug-in tree object
4. Adding the create action
5. Adding the remove action
6. Adding the alter action

Adding the folder to hold multiple objects in the tree

Procedure:

In this example, we will implement CCTreeObject instead of CCObject so that we can have plugin objects show up under Database in the Control Center tree. First

we need to create a CCTreeObject implementation for this object. It is customary to create a folder if you have multiple objects to place in the tree, rather than placing them all directly under Database. Here is an initial version of a folder:

```
  public class Example3Folder implements CCTreeObject {
    private String parentName = null;
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public CCTableObject getChildren () { return null; }
    public void setParentName(String name)
    {
      parentName = name;
    }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public CCMenuAction[] getMenuActions () { return null; }

    public String getName () { return "Example3 Folder"; }

    public void getData (Object[] data) {
       data[0] = this;
    }

    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }

    public Icon getIcon (int iconState) {
      switch (iconState) {
        case CLOSED_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);

        case OPEN_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_OPEN_
FOLDER);

        default:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);
      }
    }

  }
```

Notice that getType now makes use of a class CCTypeFactory. The purpose of CCTypeFactory is to prevent two objects from using the same type number so that the plugins can be identified as having unique types by the Control Center. Your new folder is not one of the built-in CC object types but is a new type and needs to have a new type number that must not conflict with those of any other new types you may create and must not conflict with those of the built-in types.

The getIcon method takes a parameter for iconState that lets you know if you are an open or closed folder. You can then make your icon correspond to your state, as above.

In order to show the folder in the details view when the database is selected and not just in the tree, getData needs to return a single column whose value is the plugin object itself. The getData method assigns the *this* reference to the first element of the data array. This allows both the icon and the name to appear in the same column of the details view. The Control Center, when it sees that you are returning a CCTableObject subclass, knows that it can call getIcon and getName on your Example3Folder.

The next step is to create a CCDatabase class to implement CCTreeObject and return from its getChildren method a CCTableObject array containing an instance of Example3Folder as follows:

```
import java.util.*;

  class CCDatabase implements CCTreeObject {
    private String parentName = null;
    private Vector childVector;

    public CCDatabase() {
      childVector = new Vector();
      childVector.addElement(new Example3Folder());
    }

    public CCTableObject[] getChildren() {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }

    public void setParentName(String name)
    {
      parentName = name;
    }

    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public int getType () { return UDB_DATABASE; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
  }
```

Related concepts:

- "Compiling and running the example plugins" on page 386

Related tasks:

- "Adding an example object under the folder" on page 395
- "Setting attributes for a plugin tree object" on page 397
- "Adding the create action" on page 398
- "Adding the remove action with multiple selection support" on page 400
- "Adding the alter action" on page 401

Adding an example object under the folder

Procedure:

The first step is to create a CCObject implementation for the child object as follows:

```
  class Example3Child implements CCTableObject {
    private String parentName = null;
    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
```

```
    public void setParentName(String name)
    {
      parentName = name;
    }
    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }
  }
```

Next, modify Example3Folder to keep a Vector of these Exercise3Child objects as follows:

```
  public class Example3Folder implements CCObject {
    private String parentName = null;
    private Vector childVector;
    ...

    public Example3Folder() {
      childVector = new Vector();
    }

    ...
    public CCTableObject[] getChildren () {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }
    public void setParentName(String name)
    {
      parentName = name;
    }
    ...
  }
```

For simplicity, in this example getChildren returns a array of children which are stored in the vector called childVector.

A real plugin should reconstruct the children when getChildren is called. This will refresh the list which may include new or changed child objects which may have been created or changed outside the Control Center since the last time the list was displayed. The children should be stored in and read from persistent storage so that they are not lost.

Also in a real plugin, the list of children returned by getChildren is dependent on what objects are the parents of this object in Control Center tree. The parent information is in the parentName string which is provided by the Control Center call to the setParentName method.

Note: In this example, when a refresh is done in the Control Center from the Database object or higher in the tree, the list of children under the Example3 Folder will be lost. This is because a new Example3Folder is constructed by the Control Center when the refresh is done. If this example code read the children in from persistent storage, the children would not be lost. To keep the example simple, we do not do this.

Related concepts:

- "Compiling and running the example plugins" on page 386

Related tasks:

- "Adding the folder to hold multiple objects in the tree" on page 393
- "Setting attributes for a plugin tree object" on page 397

Setting attributes for a plugin tree object

Procedure:

If you expand the tree to your plugin folder and select it, you will see that there are no columns in the details pane. This is because the Example3Child implementation of getColumns is returning null. To change this, first create some CCColumn implementations. We will create two columns because a future example will demonstrate how to change the value of one of these columns at run time and every object should have one column that should never change. We will call the unchanging column "Name" and the changing column "State".

```
class NameColumn implements CCColumn {
   getName() { return "Name"; }
   getColumnClass { return CCTableObject.class; }
}

class StateColumn implements CCColumn {
   getName() { return "State"; }
   getColumnClass { return String.class; }
}
```

The class types supported include the class equivalents of the Java primitives (such as java.lang.Intger), the java.util.Date class, and the CCTableObject class.

Change the getColumns method of Example3Child to include these two columns.

```
class Example3Child implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

You must also change the parent to include the same columns.

```
class Example3Folder implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

Now you must set the values that will be displayed for each row in the details view. You do this by setting the elements of the Object array passed into getData. The class of each column's data must match the class returned by getColumnClass for the corresponding column.

```
class Example3Child implements CCTableObject {
  ...
  private String name;
  private String state;

  public Exampe3Child(String name, String state) {
    this.name = name;
    this.state = state;
```

```
    }
    ...
    public void getData (Object[] data) {
      data[0] = this;
      data[1] = state;
    }
    ...
  }
```

In this case, the first column, which was of class CCTableObject will have a value of this. This allows the Control Center to render both the text returned by getName and the icon returned by getIcon. So the next step is to implement these. We will just use the same refresh icon used in Example 1 for the tool bar button.

```
  class Example3Child implements CCTableObject {
    ...
    public String getName () {
      return name;
    }
    public Icon getIcon () {
 return CommonImageRepository.getScaledIcon(CommonImageRepository.WC_NV_
REFRESH);
    }
    ...
  }
```

To see the results of your work so far, you can create an example child object that you will remove in the next exercise. Add an instance of Example3Child to the Example3Folder when the childVector is constructed.

```
  public class Example3Folder implements CCTreeObject {
    ...
    public Example3Folder() {
      childVector = new Vector();
      childVector.addElement(new Example3Child("Plugin1", "State1"));
    }
    ...
  }
```

Related concepts:

- "Compiling and running the example plugins" on page 386

Related tasks:

- "Adding the folder to hold multiple objects in the tree" on page 393
- "Adding an example object under the folder" on page 395
- "Adding the create action" on page 398
- "Adding the alter action" on page 401

Adding the create action

Procedure:

To allow your users to create objects under your folder at run time, you simply have to update the Vector, make your class an Observable, and call notifyObservers when the user triggers an event. The Control Center automatically registers itself as an Observer of any CCTableObjects that are Observables.

First, add a method to Example3Folder to add a child object to its vector of children.

```
public class Example3Folder implements CCTreeObject, Observable {
  ...
  public void addChild(Example3Child child) {
    childVector.addElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                     CCObjectCollectionEvent.OBJECT_ADDED,
                     child));
  }
  ...
}
```

In the above code, we are using a new class called CCObjectCollectionEvent as an argument to notifyObservers. A CCObjectCollectionEvent is an event that represents a change in a collection of CCObjects, such as a folder in the Control Center tree. The Control Center observes all CCObjects that extend Observable and responds to CCObjectCollectionEvents by updating the tree and details view. There are three types of events: add, remove, and alter.

A CCObjectCollectionEvent takes three arguments. The first is the object that triggered the event. The second is the type of event, which can be OBJECT_ADDED, OBJECT_ALTERED, or OBJECT_REMOVED. The last argument is the new object being created.

Next, add a menu item to the folder to allow the user to trigger a call to your new addChild method.

```
class CreateAction implements CCMenuAction {
  private int pluginNumber = 0;
  public String getMenuText () { return "Create"; }

  public void actionPerformed (ActionEvent e) {
Example3Folder folder = (Example3Folder)((Vector)e.getSource()).elementAt(0);
    folder.addChild(new Example3Child("Plugin " + ++pluginNumber, "State1"));
  }
}
```

The ActionEvent will always contain a Vector of all of the objects on which the action was invoked. Since this action will only be invoked on an Example3Folder and there can be only one folder, we will just cast the first object in the Vector and call addChild on it.

The last step is to add the menu action to your folder and you can now remove the sample object that was added earlier.

```
public class Example3Folder extends Observable implements CCTreeObject {
  private CCMenuAction[] menuActions =
     new CCMenuAction[] { new CreateChildAction(); }
  ...
  public Example3Folder() {
    childVector = new Vector();
  }
  ...
  public CCMenuAction[] getMenuActions () {
    retun menuActions;
  }
  ...
}
```

Related concepts:

- "Compiling and running the example plugins" on page 386

Related tasks:

- "Adding the folder to hold multiple objects in the tree" on page 393
- "Adding an example object under the folder" on page 395
- "Setting attributes for a plugin tree object" on page 397
- "Adding the remove action with multiple selection support" on page 400
- "Adding the alter action" on page 401

Adding the remove action with multiple selection support

Procedure:

Now that your users can create as many instances of your plugin as they want, you may want to give them the ability to delete as well. First, add a method to Example3Folder to remove the child and notify the Control Center.

```
public class Example3Folder extends Observable implements CCTreeObject {

  public void removeChild(Example3Child child) {
    childVector.removeElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_REMOVED,
                    child));
  }
}
```

The next step is to add a menu action to the Example3Child. We will make this CCMenuAction implement MultiSelectable so that your users can remove multiple objects at the same time. Since the source of this action will be a Vector of Example3Child objects rather than an Example3Folder, the Example3Folder should be passed in to the menu action some other way, such as in the constructor.

```
class RemoveAction implements CCMenuAction, MultiSelectable {
  private Example3Folder folder;

  public RemoveAction(Example3Folder folder) {
     this.folder = folder;
  }

  public String getMenuText () { return "Remove"; }

  public int getSelectionMode () { return MultiSelectable.MULTI_HANDLE_ONE; }

  public void actionPerformed (ActionEvent e) {
    Vector childrenVector = (Vector)e.getSource();
    for (int i = 0; i < childrenVector.size(); i++) {
      folder.removeChild((Example3Child)childrenVector.elementAt(i));
    }
  }
}
```

Implementing MultiSelectable requires you to implement getSelectionMode. In this case, it is made to return MULTI_HANDLE_ONE, which means that this menu item will appear on the menu even when multiple objects are selected and there will be a single call to your actionPerformed method for all of the selected objects.

Now add the new menu action to the Example3Child. This will involve adding a new parameter to the Example3Child constructor to pass in the folder.

```
class Example3Child implements CCTableObject {
  ...
  private CCMenuAction[] menuActions;
```

```
    public Example3Child(Example3Folder folder, String name, String state) {
      ...
      menuActions = new CCMenuAction[] { new RemoveAction(folder) };
    }
    ...
    public CCMenuAction[] getMenuActions () {
      return menuActions;
    }
  }
```

Remember to change CreateAction to use the new constructor.

```
  class CreateAction implements CCMenuAction {
    ...
    public void actionPerformed (ActionEvent e) {
      ...
 folder.addChild(new Example3Child(folder, "Plugin " + ++pluginNumber,
"State 1"));
    }
  }
```

Related concepts:

- "Compiling and running the example plugins" on page 386

Related tasks:

- "Adding the folder to hold multiple objects in the tree" on page 393
- "Adding an example object under the folder" on page 395
- "Setting attributes for a plugin tree object" on page 397
- "Adding the create action" on page 398
- "Adding the alter action" on page 401

Adding the alter action

Procedure:

The final type of event the Control Center listens to with respect to plugins is the OBJECT_ALTERED event. We created a "State" column in a previous example so that this feature could be demonstrated in this example. We will increment the state value when the Alter action is invoked.

The first step is to write a method to change the state, but this time it will be on the Example3Child rather than the folder. In this case, both the first and third arguments are the Example3Child. Remember to extend Observable.

```
  class Example3Child extends Observable implements CCTableObject {
    ...
    public void setState(String state) {
      this.state = state;
      setChanged();
      notifyObservers(new CCObjectCollectionEvent(this,
                        CCObjectCollectionEvent.OBJECT_ALTERED, this));
    }
    ...
  }
```

Next, create a menu action for Alter and add it to the CCMenuAction array in Example3Child. The AlterAction class also implements the CCDefaultMenuAction interface to define Alter as the default action which gets invoked when the user double clicks on an Example3Child object in the Control Center.

```
class AlterAction implements CCMenuAction, CCDefaultMenuAction {
  private int stateNumber = 1;
  public String getMenuText () { return "Alter"; }

  public void actionPerformed (ActionEvent e) {
    ((Example3Child)((Vector)e.getSource()).elementAt(0)).setState("State "
        + ++stateNumber);
  }
}

class Example3Child implements CCTableObject {
  ...
  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new AlterAction(),
                                       new RemoveAction(folder) };
  }
  ...
}
```

Related concepts:

- "Compiling and running the example plugins" on page 386

Related tasks:

- "Adding the folder to hold multiple objects in the tree" on page 393
- "Adding an example object under the folder" on page 395
- "Setting attributes for a plugin tree object" on page 397
- "Adding the create action" on page 398
- "Adding the remove action with multiple selection support" on page 400

Disabling configuration features with isConfigurable()

Procedure:

Returning a value of false in the isConfigurable method of a CCObject of type UDB_DATABASE or UDB_INSTANCE will remove the Configure menu item from the Database and Instance popup menu respectively.

Related concepts:

- "Compiling and running the example plugins" on page 386

Disabling the ability to alter objects using isEditable()

Procedure:

Any of the Control Center objects that support an Alter action can have that action removed by creating a plugin for that object and returning false from the isEditable method. You can also specify that the Alter action is removed only for an object whose fully qualified name matches the value returned by the getName method for that object.

An additional feature is to add a Browse action. This can be done for UDB Tables, Views, and Indexes only. The Browse action is added by putting a file in db2plug.zip called BrowseTable, BrowseView or BrowseIndex. The files can be empty but their names must be BrowseTable, BrowseView or BrowseIndex. The addition of a Browse button cannot be restricted to an object with a particular name.

Related concepts:

- "Compiling and running the example plugins" on page 386

Disabling the default buttons in configuration dialogs using hasConfigurationDefaults()

Procedure:

The configuration dialogs for UDB databases and instances contain buttons for settings the values back to the DB2 defaults. If you have your own set of defaults and do not want users mistakenly hitting these buttons, you can disable the buttons using a plugin. Create an object that implements CCObject and set its type to UDB_DATABASE or UDB_INSTANCE. Have it also implement the Defaultable interface. This interface contains a method called hasConfigurationDefaults. Return false from this method and all default buttons for the configuration dialog will be disabled. The example disables the default buttons for UDB Database Configuration.

Related concepts:

- "Compiling and running the example plugins" on page 386

Appendix K. DB2 Universal Database technical information

DB2 documentation and help

DB2® technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics
  - Help for DB2 tools
  - Sample programs
  - Tutorials
- Downloadable PDF files, PDF files on CD, and printed books
  - Guides
  - Reference manuals
- Command line help
  - Command help
  - Message help
  - SQL state help
- Installed source code
  - Sample programs

You can access additional DB2 Universal Database™ technical information such as technotes, white papers, and Redbooks™ online at ibm.com®. Access the DB2 Information Management software library site at www.ibm.com/software/data/pubs/.

DB2 documentation updates

IBM® may periodically make documentation FixPaks and other documentation updates to the DB2 Information Center available. If you access the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2help/, you will always be viewing the most up-to-date information. If you have installed the DB2 Information Center locally, then you need to install any updates manually before you can view them. Documentation updates allow you to update the information that you installed from the *DB2 Information Center CD* when new information becomes available.

The Information Center is updated more frequently than either the PDF or the hardcopy books. To get the most current DB2 technical information, install the documentation updates as they become available or go to the DB2 Information Center at the www.ibm.com site.

Related concepts:

- "CLI sample programs" in the *CLI Guide and Reference, Volume 1*
- "Java sample programs" in the *Application Development Guide: Building and Running Applications*
- "DB2 Information Center" on page 406

Related tasks:

- "Invoking contextual help from a DB2 tool" on page 423

- "Updating the DB2 Information Center installed on your computer or intranet server" on page 415
- "Invoking message help from the command line processor" on page 424
- "Invoking command help from the command line processor" on page 425
- "Invoking SQL state help from the command line processor" on page 425

Related reference:

- "DB2 PDF and printed documentation" on page 417

DB2 Information Center

The DB2® Information Center gives you access to all of the information you need to take full advantage of DB2 family products, including DB2 Universal Database™, DB2 Connect™, DB2 Information Integrator and DB2 Query Patroller™. The DB2 Information Center also contains information for major DB2 features and components including replication, data warehousing, and the DB2 extenders.

The DB2 Information Center has the following features if you view it in Mozilla 1.0 or later or Microsoft® Internet Explorer 5.5 or later. Some features require you to enable support for JavaScript™:

Flexible installation options

You can choose to view the DB2 documentation using the option that best meets your needs:

- To effortlessly ensure that your documentation is always up to date, you can access all of your documentation directly from the DB2 Information Center hosted on the IBM® Web site at http://publib.boulder.ibm.com/infocenter/db2help/
- To minimize your update efforts and keep your network traffic within your intranet, you can install the DB2 documentation on a single server on your intranet
- To maximize your flexibility and reduce your dependence on network connections, you can install the DB2 documentation on your own computer

Search

You can search all of the topics in the DB2 Information Center by entering a search term in the **Search** text field. You can retrieve exact matches by enclosing terms in quotation marks, and you can refine your search with wildcard operators (*, ?) and Boolean operators (AND, NOT, OR).

Task-oriented table of contents

You can locate topics in the DB2 documentation from a single table of contents. The table of contents is organized primarily by the kind of tasks you may want to perform, but also includes entries for product overviews, goals, reference information, an index, and a glossary.

- Product overviews describe the relationship between the available products in the DB2 family, the features offered by each of those products, and up to date release information for each of these products.
- Goal categories such as installing, administering, and developing include topics that enable you to quickly complete tasks and develop a deeper understanding of the background information for completing those tasks.

- Reference topics provide detailed information about a subject, including statement and command syntax, message help, and configuration parameters.

Show current topic in table of contents
You can show where the current topic fits into the table of contents by clicking the **Refresh / Show Current Topic** button in the table of contents frame or by clicking the **Show in Table of Contents** button in the content frame. This feature is helpful if you have followed several links to related topics in several files or arrived at a topic from search results.

Index You can access all of the documentation from the index. The index is organized in alphabetical order by index term.

Glossary
You can use the glossary to look up definitions of terms used in the DB2 documentation. The glossary is organized in alphabetical order by glossary term.

Integrated localized information
The DB2 Information Center displays information in the preferred language set in your browser preferences. If a topic is not available in your preferred language, the DB2 Information Center displays the English version of that topic.

For iSeries™ technical information, refer to the IBM eServer™ iSeries information center at www.ibm.com/eserver/iseries/infocenter/.

Related concepts:

- "DB2 Information Center installation scenarios" on page 407

Related tasks:

- "Updating the DB2 Information Center installed on your computer or intranet server" on page 415
- "Displaying topics in your preferred language in the DB2 Information Center" on page 416
- "Invoking the DB2 Information Center" on page 414
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 410
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 412

DB2 Information Center installation scenarios

Different working environments can pose different requirements for how to access DB2® information. The DB2 Information Center can be accessed on the IBM® Web site, on a server on your organization's network, or on a version installed on your computer. In all three cases, the documentation is contained in the DB2 Information Center, which is an architected web of topic-based information that you view with a browser. By default, DB2 products access the DB2 Information Center on the IBM Web site. However, if you want to access the DB2 Information Center on an intranet server or on your own computer, you must install the DB2 Information Center using the DB2 Information Center CD found in your product Media Pack. Refer to the summary of options for accessing DB2 documentation which follows, along with the three installation scenarios, to help determine which

method of accessing the DB2 Information Center works best for you and your work environment, and what installation issues you might need to consider.

Summary of options for accessing DB2 documentation:

The following table provides recommendations on which options are possible in your work environment for accessing the DB2 product documentation in the DB2 Information Center.

Internet access	Intranet access	Recommendation
Yes	Yes	Access the DB2 Information Center on the IBM Web site, or access the DB2 Information Center installed on an intranet server.
Yes	No	Access the DB2 Information Center on the IBM Web site.
No	Yes	Access the DB2 Information Center installed on an intranet server.
No	No	Access the DB2 Information Center on a local computer.

Scenario: Accessing the DB2 Information Center on your computer:

Tsu-Chen owns a factory in a small town that does not have a local ISP to provide him with Internet access. He purchased DB2 Universal Database™ to manage his inventory, his product orders, his banking account information, and his business expenses. Never having used a DB2 product before, Tsu-Chen needs to learn how to do so from the DB2 product documentation.

After installing DB2 Universal Database on his computer using the typical installation option, Tsu-Chen tries to access the DB2 documentation. However, his browser gives him an error message that the page he tried to open cannot be found. Tsu-Chen checks the installation manual for his DB2 product and discovers that he has to install the DB2 Information Center if he wants to access DB2 documentation on his computer. He finds the *DB2 Information Center CD* in the media pack and installs it.

From the application launcher for his operating system, Tsu-Chen now has access to the DB2 Information Center and can learn how to use his DB2 product to increase the success of his business.

Scenario: Accessing the DB2 Information Center on the IBM Web site:

Colin is an information technology consultant with a training firm. He specializes in database technology and SQL and gives seminars on these subjects to businesses all over North America using DB2 Universal Database. Part of Colin's seminars includes using DB2 documentation as a teaching tool. For example, while teaching courses on SQL, Colin uses the DB2 documentation on SQL as a way to teach basic and advanced syntax for database queries.

Most of the businesses at which Colin teaches have Internet access. This situation influenced Colin's decision to configure his mobile computer to access the DB2 Information Center on the IBM Web site when he installed the latest version of DB2 Universal Database. This configuration allows Colin to have online access to the latest DB2 documentation during his seminars.

However, sometimes while travelling Colin does not have Internet access. This posed a problem for him, especially when he needed to access to DB2 documentation to prepare for seminars. To avoid situations like this, Colin installed a copy of the DB2 Information Center on his mobile computer.

Colin enjoys the flexibility of always having a copy of DB2 documentation at his disposal. Using the **db2set** command, he can easily configure the registry variables on his mobile computer to access the DB2 Information Center on either the IBM Web site, or his mobile computer, depending on his situation.

Scenario: Accessing the DB2 Information Center on an intranet server:

Eva works as a senior database administrator for a life insurance company. Her administration responsibilities include installing and configuring the latest version of DB2 Universal Database on the company's UNIX® database servers. Her company recently informed its employees that, for security reasons, it would not provide them with Internet access at work. Because her company has a networked environment, Eva decides to install a copy of the DB2 Information Center on an intranet server so that all employees in the company who use the company's data warehouse on a regular basis (sales representatives, sales managers, and business analysts) have access to DB2 documentation.

Eva instructs her database team to install the latest version of DB2 Universal Database on all of the employee's computers using a response file, to ensure that each computer is configured to access the DB2 Information Center using the host name and the port number of the intranet server.

However, through a misunderstanding Migual, a junior database administrator on Eva's team, installs a copy of the DB2 Information Center on several of the employee computers, rather than configuring DB2 Universal Database to access the DB2 Information Center on the intranet server. To correct this situation Eva tells Migual to use the **db2set** command to change the DB2 Information Center registry variables (DB2_DOCHOST for the host name, and DB2_DOCPORT for the port number) on each of these computers. Now all of the appropriate computers on the network have access to the DB2 Information Center, and employees can find answers to their DB2 questions in the DB2 documentation.

Related concepts:

- "DB2 Information Center" on page 406

Related tasks:

- "Updating the DB2 Information Center installed on your computer or intranet server" on page 415
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 410
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 412
- "Setting the location for accessing the DB2 Information Center: Common GUI help"

Related reference:

- "db2set - DB2 Profile Registry Command" in the *Command Reference*

Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)

DB2 product documentation can be accessed in three ways: on the IBM Web site, on an intranet server, or on a version installed on your computer. By default, DB2 products access DB2 documentation on the IBM Web site. If you want to access the DB2 documentation on an intranet server or on your own computer, you must install the documentation from the *DB2 Information Center CD*. Using the DB2 Setup wizard, you can define your installation preferences and install the DB2 Information Center on a computer that uses a UNIX operating system.

Prerequisites:

This section lists the hardware, operating system, software, and communication requirements for installing the DB2 Information Center on UNIX computers.

- **Hardware requirements**

  You require one of the following processors:
  - PowerPC (AIX)
  - HP 9000 (HP-UX)
  - Intel 32–bit (Linux)
  - Solaris UltraSPARC computers (Solaris Operating Environment)
- **Operating system requirements**

  You require one of the following operating systems:
  - IBM AIX 5.1 (on PowerPC)
  - HP-UX 11i (on HP 9000)
  - Red Hat Linux 8.0 (on Intel 32–bit)
  - SuSE Linux 8.1 (on Intel 32–bit)
  - Sun Solaris Version 8 (on Solaris Operating Environment UltraSPARC computers)

  Note: The DB2 Information Center runs on a subset of the UNIX operating systems on which DB2 clients are supported. It is therefore recommended that you either access the DB2 Information Center from the IBM Web site, or that you install and access the DB2 Information Center on an intranet server.
- **Software requirements**
  - The following browser is supported:
    - Mozilla Version 1.0 or greater
- The DB2 Setup wizard is a graphical installer. You must have an implementation of the X Window System software capable of rendering a graphical user interface for the DB2 Setup wizard to run on your computer. Before you can run the DB2 Setup wizard you must ensure that you have properly exported your display. For example, enter the following command at the command prompt:

  ```
  export DISPLAY=9.26.163.144:0.
  ```
- **Communication requirements**
  - TCP/IP

Procedure:

To install the DB2 Information Center using the DB2 Setup wizard:

1. Log on to the system.
2. Insert and mount the DB2 Information Center product CD on your system.
3. Change to the directory where the CD is mounted by entering the following command:

   ```
   cd /cd
   ```

   where */cd* represents the mount point of the CD.
4. Enter the **./db2setup** command to start the DB2 Setup wizard.
5. The IBM DB2 Setup Launchpad opens. To proceed directly to the installation of the DB2 Information Center, click **Install Product**. Online help is available to guide you through the remaining steps. To invoke the online help, click **Help**. You can click **Cancel** at any time to end the installation.
6. On the **Select the product you would like to install** page, click **Next**.
7. Click **Next** on the **Welcome to the DB2 Setup wizard** page. The DB2 Setup wizard will guide you through the program setup process.
8. To proceed with the installation, you must accept the license agreement. On the **License Agreement** page, select **I accept the terms in the license agreement** and click **Next**.
9. Select **Install DB2 Information Center on this computer** on the **Select the installation action** page. If you want to use a response file to install the DB2 Information Center on this or other computers at a later time, select **Save your settings in a response file**. Click **Next**.
10. Select the languages in which the DB2 Information Center will be installed on **Select the languages to install** page. Click **Next**.
11. Configure the DB2 Information Center for incoming communication on the **Specify the DB2 Information Center port** page. Click **Next** to continue the installation.
12. Review the installation choices you have made in the **Start copying files** page. To change any settings, click **Back**. Click **Install** to copy the DB2 Information Center files onto your computer.

You can also install the DB2 Information Center using a response file.

The installation logs `db2setup.his`, `db2setup.log`, and `db2setup.err` are located, by default, in the `/tmp` directory.

The `db2setup.log` file captures all DB2 product installation information, including errors. The `db2setup.his` file records all DB2 product installations on your computer. DB2 appends the `db2setup.log` file to the `db2setup.his` file. The `db2setup.err` file captures any error output that is returned by Java, for example, exceptions and trap information.

When the installation is complete, the DB2 Information Center will be installed in one of the following directories, depending upon your UNIX operating system:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris Operating Environment: /opt/IBM/db2/V8.1

Related concepts:

- "DB2 Information Center" on page 406
- "DB2 Information Center installation scenarios" on page 407

Related tasks:

- "Installing DB2 using a response file (UNIX)" in the *Installation and Configuration Supplement*
- "Updating the DB2 Information Center installed on your computer or intranet server" on page 415
- "Displaying topics in your preferred language in the DB2 Information Center" on page 416
- "Invoking the DB2 Information Center" on page 414
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 412

Installing the DB2 Information Center using the DB2 Setup wizard (Windows)

DB2 product documentation can be accessed in three ways: on the IBM Web site, on an intranet server, or on a version installed on your computer. By default, DB2 products access DB2 documentation on the IBM Web site. If you want to access the DB2 documentation on an intranet server or on your own computer, you must install the DB2 documentation from the *DB2 Information Center CD*. Using the DB2 Setup wizard, you can define your installation preferences and install the DB2 Information Center on a computer that uses a Windows operating system.

Prerequisites:

This section lists the hardware, operating system, software, and communication requirements for installing the DB2 Information Center on Windows.

- **Hardware requirements**

  You require one of the following processors:

  – 32-bit computers: a Pentium or Pentium compatible CPU
- **Operating system requirements**

  You require one of the following operating systems:

  – Windows 2000

  – Windows XP

  Note: The DB2 Information Center runs on a subset of the Windows operating systems on which DB2 clients are supported. It is therefore recommended that you either access the DB2 Information Center on the IBM Web site, or that you install and access the DB2 Information Center on an intranet server.
- **Software requirements**

  – The following browsers are supported:

    - Mozilla 1.0 or greater

    - Internet Explorer Version 5.5 or 6.0 (Version 6.0 for Windows XP)
- **Communication requirements**

  – TCP/IP

Restrictions:

- You require an account with administrative privileges to install the DB2 Information Center.

Procedure:

To install the DB2 Information Center using the DB2 Setup wizard:

1. Log on to the system with the account that you have defined for the DB2 Information Center installation.
2. Insert the CD into the drive. If enabled, the auto-run feature starts the IBM DB2 Setup Launchpad.
3. The DB2 Setup wizard determines the system language and launches the setup program for that language. If you want to run the setup program in a language other than English, or the setup program fails to auto-start, you can start the DB2 Setup wizard manually.

   To start the DB2 Setup wizard manually:

   a. Click **Start** and select **Run**.
   b. In the **Open** field, type the following command:

      ```
      x:\setup.exe /i 2-letter language identifier
      ```

      where *x:* represents your CD drive, and *2-letter language identifier* represents the language in which the setup program will be run.
   c. Click **OK**.
4. The IBM DB2 Setup Launchpad opens. To proceed directly to the installation of the DB2 Information Center, click **Install Product**. Online help is available to guide you through the remaining steps. To invoke the online help, click **Help**. You can click **Cancel** at any time to end the installation.
5. On the **Select the product you would like to install** page, click **Next**.
6. Click **Next** on the **Welcome to the DB2 Setup wizard** page. The DB2 Setup wizard will guide you through the program setup process.
7. To proceed with the installation, you must accept the license agreement. On the **License Agreement** page, select **I accept the terms in the license agreement** and click **Next**.
8. Select **Install DB2 Information Center on this computer** on the **Select the installation action** page. If you want to use a response file to install the DB2 Information Center on this or other computers at a later time, select **Save your settings in a response file**. Click **Next**.
9. Select the languages in which the DB2 Information Center will be installed on **Select the languages to install** page. Click **Next**.
10. Configure the DB2 Information Center for incoming communication on the **Specify the DB2 Information Center port** page. Click **Next** to continue the installation.
11. Review the installation choices you have made in the **Start copying files** page. To change any settings, click **Back**. Click **Install** to copy the DB2 Information Center files onto your computer.

You can install the DB2 Information Center using a response file. You can also use the **db2rspgn** command to generate a response file based on an existing installation.

For information on errors encountered during installation, see the `db2.log` and `db2wi.log` files located in the `'My Documents'\DB2LOG\` directory. The location of the `'My Documents'` directory will depend on the settings on your computer.

The `db2wi.log` file captures the most recent DB2 installation information. The `db2.log` captures the history of DB2 product installations.

Related concepts:

- "DB2 Information Center" on page 406
- "DB2 Information Center installation scenarios" on page 407

Related tasks:

- "Installing a DB2 product using a response file (Windows)" in the *Installation and Configuration Supplement*
- "Updating the DB2 Information Center installed on your computer or intranet server" on page 415
- "Displaying topics in your preferred language in the DB2 Information Center" on page 416
- "Invoking the DB2 Information Center" on page 414
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 410

Related reference:

- "db2rspgn - Response File Generator Command (Windows)" in the *Command Reference*

Invoking the DB2 Information Center

The DB2 Information Center gives you access to all of the information that you need to use DB2 products for Linux, UNIX, and Windows operating systems such as DB2 Universal Database, DB2 Connect, DB2 Information Integrator, and DB2 Query Patroller.

You can invoke the DB2 Information Center from one of the following places:

- Computers on which a DB2 UDB client or server is installed
- An intranet server or local computer on which the DB2 Information Center installed
- The IBM Web site

Prerequisites:

Before you invoke the DB2 Information Center:

- *Optional*: Configure your browser to display topics in your preferred language
- *Optional*: Configure your DB2 client to use the DB2 Information Center installed on your computer or intranet server

Procedure:

To invoke the DB2 Information Center on a computer on which a DB2 UDB client or server is installed:

- From the Start Menu (Windows operating system): Click **Start → Programs → IBM DB2 → Information → Information Center**.
- From the command line prompt:
  - For Linux and UNIX operating systems, issue the **db2icdocs** command.
  - For the Windows operating system, issue the **db2icdocs.exe** command.

To open the DB2 Information Center installed on an intranet server or local computer in a Web browser:

- Open the Web page at http://<host-name>:<port-number>/, where <host-name> represents the host name and <port-number> represents the port number on which the DB2 Information Center is available.

To open the DB2 Information Center on the IBM Web site in a Web browser:

- Open the Web page at publib.boulder.ibm.com/infocenter/db2help/.

Related concepts:

- "DB2 Information Center" on page 406
- "DB2 Information Center installation scenarios" on page 407

Related tasks:

- "Invoking contextual help from a DB2 tool" on page 423
- "Updating the DB2 Information Center installed on your computer or intranet server" on page 415
- "Invoking command help from the command line processor" on page 425
- "Setting the location for accessing the DB2 Information Center: Common GUI help"

Related reference:

- "HELP Command" in the *Command Reference*

Updating the DB2 Information Center installed on your computer or intranet server

The DB2 Information Center available from http://publib.boulder.ibm.com/infocenter/db2help/ will be periodically updated with new or changed documentation. IBM may also make DB2 Information Center updates available to download and install on your computer or intranet server. Updating the DB2 Information Center does not update DB2 client or server products.

Prerequisites:

You must have access to a computer that is connected to the Internet.

Procedure:

To update the DB2 Information Center installed on your computer or intranet server:

1. Open the DB2 Information Center hosted on the IBM Web site at: http://publib.boulder.ibm.com/infocenter/db2help/
2. In the Downloads section of the welcome page under the Service and Support heading, click the **DB2 Universal Database documentation** link.
3. Determine if the version of your DB2 Information Center is out of date by comparing the latest refreshed documentation image level to the documentation level you have installed. The documentation level you have installed is listed on the DB2 Information Center welcome page.
4. If a more recent version of the DB2 Information Center is available, download the latest refreshed *DB2 Information Center* image applicable to your operating system.

5. To install the refreshed *DB2 Information Center* image, follow the instructions provided on the Web page.

Related concepts:

- "DB2 Information Center installation scenarios" on page 407

Related tasks:

- "Invoking the DB2 Information Center" on page 414
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 410
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 412

Displaying topics in your preferred language in the DB2 Information Center

The DB2 Information Center attempts to display topics in the language specified in your browser preferences. If a topic has not been translated into your preferred language, the DB2 Information Center displays the topic in English.

Procedure:

To display topics in your preferred language in the Internet Explorer browser:

1. In Internet Explorer, click the **Tools** —> **Internet Options** —> **Languages...** button. The Language Preferences window opens.
2. Ensure your preferred language is specified as the first entry in the list of languages.
   - To add a new language to the list, click the **Add...** button.

     Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.
   - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Refresh the page to display the DB2 Information Center in your preferred language.

To display topics in your preferred language in the Mozilla browser:

1. In Mozilla, select the **Edit** —> **Preferences** —> **Languages** button. The Languages panel is displayed in the Preferences window.
2. Ensure your preferred language is specified as the first entry in the list of languages.
   - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
   - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Refresh the page to display the DB2 Information Center in your preferred language.

Related concepts:

- "DB2 Information Center" on page 406

DB2 PDF and printed documentation

The following tables provide official book names, form numbers, and PDF file names. To order hardcopy books, you must know the official book name. To print a PDF file, you must know the PDF file name.

The DB2 documentation is categorized by the following headings:

- Core DB2 information
- Administration information
- Application development information
- Business intelligence information
- DB2 Connect information
- Getting started information
- Tutorial information
- Optional component information
- Release notes

The following tables describe, for each book in the DB2 library, the information needed to order the hard copy, or to print or view the PDF for that book. A full description of each of the books in the DB2 library is available from the IBM Publications Center at www.ibm.com/shop/publications/order

Core DB2 information

The information in these books is fundamental to all DB2 users; you will find this information useful whether you are a programmer, a database administrator, or someone who works with DB2 Connect, DB2 Warehouse Manager, or other DB2 products.

Table 53. Core DB2 information

Name	Form Number	PDF File Name
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x81
IBM DB2 Universal Database Glossary	No form number	db2t0x81
IBM DB2 Universal Database Message Reference, Volume 1	GC09-4840, not available in hardcopy	db2m1x81
IBM DB2 Universal Database Message Reference, Volume 2	GC09-4841, not available in hardcopy	db2m2x81
IBM DB2 Universal Database What's New	SC09-4848	db2q0x81

Administration information

The information in these books covers those topics required to effectively design, implement, and maintain DB2 databases, data warehouses, and federated systems.

Table 54. Administration information

Name	Form number	PDF file name
IBM DB2 Universal Database Administration Guide: Planning	SC09-4822	db2d1x81

Table 54. Administration information (continued)

Name	Form number	PDF file name
IBM DB2 Universal Database Administration Guide: Implementation	SC09-4820	db2d2x81
IBM DB2 Universal Database Administration Guide: Performance	SC09-4821	db2d3x81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x81
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx81
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference	SC09-4831	db2hax81
IBM DB2 Universal Database Data Warehouse Center Administration Guide	SC27-1123	db2ddx81
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x81
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x81

Application development information

The information in these books is of special interest to application developers or programmers working with DB2 Universal Database (DB2 UDB). You will find information about supported languages and compilers, as well as the documentation required to access DB2 UDB using the various supported programming interfaces, such as embedded SQL, ODBC, JDBC, SQLJ, and CLI. If you are using the DB2 Information Center, you can also access HTML versions of the source code for the sample programs.

Table 55. Application development information

Name	Form number	PDF file name
IBM DB2 Universal Database Application Development Guide: Building and Running Applications	SC09-4825	db2axx81
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x81
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x81

Table 55. Application development information (continued)

Name	Form number	PDF file name
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx81
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx81

Business intelligence information

The information in these books describes how to use components that enhance the data warehousing and analytical capabilities of DB2 Universal Database.

Table 56. Business intelligence information

Name	Form number	PDF file name
IBM DB2 Warehouse Manager Standard Edition Information Catalog Center Administration Guide	SC27-1125	db2dix81
IBM DB2 Warehouse Manager Standard Edition Installation Guide	GC27-1122	db2idx81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mstx80

DB2 Connect information

The information in this category describes how to access data on mainframe and midrange servers using DB2 Connect Enterprise Edition or DB2 Connect Personal Edition.

Table 57. DB2 Connect information

Name	Form number	PDF file name
IBM Connectivity Supplement	No form number	db2h1x81
IBM DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition	GC09-4833	db2c6x81
IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition	GC09-4834	db2c1x81
IBM DB2 Connect User's Guide	SC09-4835	db2c0x81

Getting started information

The information in this category is useful when you are installing and configuring servers, clients, and other DB2 products.

Table 58. Getting started information

Name	Form number	PDF file name
IBM DB2 Universal Database Quick Beginnings for DB2 Clients	GC09-4832, not available in hardcopy	db2itx81
IBM DB2 Universal Database Quick Beginnings for DB2 Servers	GC09-4836	db2isx81
IBM DB2 Universal Database Quick Beginnings for DB2 Personal Edition	GC09-4838	db2i1x81
IBM DB2 Universal Database Installation and Configuration Supplement	GC09-4837, not available in hardcopy	db2iyx81
IBM DB2 Universal Database Quick Beginnings for DB2 Data Links Manager	GC09-4829	db2z6x81

Tutorial information

Tutorial information introduces DB2 features and teaches how to perform various tasks.

Table 59. Tutorial information

Name	Form number	PDF file name
Business Intelligence Tutorial: Introduction to the Data Warehouse	No form number	db2tux81
Business Intelligence Tutorial: Extended Lessons in Data Warehousing	No form number	db2tax81
Information Catalog Center Tutorial	No form number	db2aix81
Video Central for e-business Tutorial	No form number	db2twx81
Visual Explain Tutorial	No form number	db2tvx81

Optional component information

The information in this category describes how to work with optional DB2 components.

Table 60. Optional component information

Name	Form number	PDF file name
IBM DB2 Cube Views Guide and Reference	SC18–7298	db2aax81
IBM DB2 Query Patroller Guide: Installation, Administration and Usage Guide	GC09–7658	db2dwx81
IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference	SC27-1226	db2sbx81

Table 60. Optional component information (continued)

Name	Form number	PDF file name
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x82
DB2 Net Search Extender Administration and User's Guide **Note:** HTML for this document is *not* installed from the HTML documentation CD.	SH12-6740	N/A

Release notes

The release notes provide additional information specific to your product's release and FixPak level. The release notes also provide summaries of the documentation updates incorporated in each release, update, and FixPak.

Table 61. Release notes

Name	Form number	PDF file name
DB2 Release Notes	See note.	See note.
DB2 Installation Notes	Available on product CD-ROM only.	Not available.

Note: The Release Notes are available in:

- XHTML and Text format, on the product CDs
- PDF format, on the PDF Documentation CD

In addition the portions of the Release Notes that discuss *Known Problems and Workarounds* and *Incompatibilities Between Releases* also appear in the DB2 Information Center.

To view the Release Notes in text format 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:

- For AIX operating systems: `/usr/opt/db2_08_01`
- For all other UNIX-based operating systems: `/opt/IBM/db2/V8.1`

Related concepts:

- "DB2 documentation and help" on page 405

Related tasks:

- "Printing DB2 books from PDF files" on page 422
- "Ordering printed DB2 books" on page 422
- "Invoking contextual help from a DB2 tool" on page 423

Printing DB2 books from PDF files

You can print DB2 books from the PDF files on the *DB2 PDF Documentation* CD. Using Adobe Acrobat Reader, you can print either the entire book or a specific range of pages.

Prerequisites:

Ensure that you have Adobe Acrobat Reader installed. If you need to install Adobe Acrobat Reader, it is available from the Adobe Web site at www.adobe.com

Procedure:

To print a DB2 book from a PDF file:

1. Insert the *DB2 PDF Documentation* CD. On UNIX operating systems, mount the DB2 PDF Documentation CD. Refer to your *Quick Beginnings* book for details on how to mount a CD on UNIX operating systems.
2. Open index.htm. The file opens in a browser window.
3. Click on the title of the PDF you want to see. The PDF will open in Acrobat Reader.
4. Select **File → Print** to print any portions of the book that you want.

Related concepts:

- "DB2 Information Center" on page 406

Related tasks:

- "Mounting the CD-ROM (AIX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (HP-UX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (Linux)" in the *Quick Beginnings for DB2 Servers*
- "Ordering printed DB2 books" on page 422
- "Mounting the CD-ROM (Solaris Operating Environment)" in the *Quick Beginnings for DB2 Servers*

Related reference:

- "DB2 PDF and printed documentation" on page 417

Ordering printed DB2 books

If you prefer to use hardcopy books, you can order them in one of three ways.

Procedure:

Printed books can be ordered in some countries or regions. Check the IBM Publications website for your country or region to see if this service is available in your country or region. When the publications are available for ordering, you can:

- Contact your IBM authorized dealer or marketing representative. To find a local IBM representative, check the IBM Worldwide Directory of Contacts at www.ibm.com/planetwide
- Phone `1-800-879-2755` in the United States or `1-800-IBM-4YOU` in Canada.

- Visit the IBM Publications Center at http://www.ibm.com/shop/publications/order. The ability to order books from the IBM Publications Center may not be available in all countries.

At the time the DB2 product becomes available, the printed books are the same as those that are available in PDF format on the *DB2 PDF Documentation CD*. Content in the printed books that appears in the *DB2 Information Center CD* is also the same. However, there is some additional content available in DB2 Information Center CD that does not appear anywhere in the PDF books (for example, SQL Administration routines and HTML samples). Not all books available on the DB2 PDF Documentation CD are available for ordering in hardcopy.

Note: The DB2 Information Center is updated more frequently than either the PDF or the hardcopy books; install documentation updates as they become available or refer to the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2help/ to get the most current information.

Related tasks:

- "Printing DB2 books from PDF files" on page 422

Related reference:

- "DB2 PDF and printed documentation" on page 417

Invoking contextual help from a DB2 tool

Contextual help provides information about the tasks or controls that are associated with a particular window, notebook, wizard, or advisor. Contextual help is available from DB2 administration and development tools that have graphical user interfaces. There are two types of contextual help:

- Help accessed through the **Help** button that is located on each window or notebook
- Infopops, which are pop-up information windows displayed when the mouse cursor is placed over a field or control, or when a field or control is selected in a window, notebook, wizard, or advisor and F1 is pressed.

The **Help** button gives you access to overview, prerequisite, and task information. The infopops describe the individual fields and controls.

Procedure:

To invoke contextual help:

- For window and notebook help, start one of the DB2 tools, then open any window or notebook. Click the **Help** button at the bottom right corner of the window or notebook to invoke the contextual help.

  You can also access the contextual help from the **Help** menu item at the top of each of the DB2 tools centers.

  Within wizards and advisors, click on the Task Overview link on the first page to view contextual help.
- For infopop help about individual controls on a window or notebook, click the control, then click **F1**. Pop-up information containing details about the control is displayed in a yellow window.

Note: To display infopops simply by holding the mouse cursor over a field or control, select the **Automatically display infopops** check box on the **Documentation** page of the Tool Settings notebook.

Similar to infopops, diagnosis pop-up information is another form of context-sensitive help; they contain data entry rules. Diagnosis pop-up information is displayed in a purple window that appears when data that is not valid or that is insufficient is entered. Diagnosis pop-up information can appear for:

- Compulsory fields.
- Fields whose data follows a precise format, such as a date field.

Related tasks:

- "Invoking the DB2 Information Center" on page 414
- "Invoking message help from the command line processor" on page 424
- "Invoking command help from the command line processor" on page 425
- "Invoking SQL state help from the command line processor" on page 425
- "Access to the DB2 Information Center: Concepts help"
- "How to use the DB2 UDB help: Common GUI help"
- "Setting the location for accessing the DB2 Information Center: Common GUI help"
- "Setting up access to DB2 contextual help and documentation: Common GUI help"

Invoking message help from the command line processor

Message help describes the cause of a message and describes any action you should take in response to the error.

Procedure:

To invoke message help, open the command line processor and enter:

```
? XXXnnnnn
```

where *XXXnnnnn* represents a valid message identifier.

For example, `? SQL30081` displays help about the SQL30081 message.

Related concepts:

- "Introduction to messages" in the *Message Reference Volume 1*

Related reference:

- "db2 - Command Line Processor Invocation Command" in the *Command Reference*

Invoking command help from the command line processor

Command help explains the syntax of commands in the command line processor.

Procedure:

To invoke command help, open the command line processor and enter:

```
? command
```

where *command* represents a keyword or the entire command.

For example, `? catalog` displays help for all of the CATALOG commands, while `? catalog database` displays help only for the CATALOG DATABASE command.

Related tasks:

- "Invoking contextual help from a DB2 tool" on page 423
- "Invoking the DB2 Information Center" on page 414
- "Invoking message help from the command line processor" on page 424
- "Invoking SQL state help from the command line processor" on page 425

Related reference:

- "db2 - Command Line Processor Invocation Command" in the *Command Reference*

Invoking SQL state help from the command line processor

DB2 Univerrsal Database returns an SQLSTATE value for conditions that could be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

Procedure:

To invoke SQL state help, open the command line processor and 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, and `? 08` displays help for the 08 class code.

Related tasks:

- "Invoking the DB2 Information Center" on page 414
- "Invoking message help from the command line processor" on page 424
- "Invoking command help from the command line processor" on page 425

DB2 tutorials

The DB2® tutorials help you learn about various aspects of DB2 Universal Database. The tutorials provide lessons with step-by-step instructions in the areas of developing applications, tuning SQL query performance, working with data warehouses, managing metadata, and developing Web services using DB2.

Before you begin:

You can view the XHTML versions of the tutorials from the Information Center at http://publib.boulder.ibm.com/infocenter/db2help/.

Some tutorial lessons use sample data or code. See each tutorial for a description of any prerequisites for its specific tasks.

DB2 Universal Database tutorials:

Click on a tutorial title in the following list to view that tutorial.

Business Intelligence Tutorial: Introduction to the Data Warehouse Center
Perform introductory data warehousing tasks using the Data Warehouse Center.

Business Intelligence Tutorial: Extended Lessons in Data Warehousing
Perform advanced data warehousing tasks using the Data Warehouse Center.

Information Catalog Center Tutorial
Create and manage an information catalog to locate and use metadata using the Information Catalog Center.

Visual Explain Tutorial
Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2® products.

DB2 documentation
Troubleshooting information can be found throughout the DB2 Information Center, as well as throughout the PDF books that make up the DB2 library. You can refer to the "Support and troubleshooting" branch of the DB2 Information Center navigation tree (in the left pane of your browser window) to see a complete listing of the DB2 troubleshooting documentation.

DB2 Technical Support Web site
Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs), FixPaks and the latest listing of internal DB2 error codes, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at http://www.ibm.com/software/data/db2/udb/winos2unix/support

DB2 Problem Determination Tutorial Series
Refer to the DB2 Problem Determination Tutorial Series Web site to find information on how to quickly identify and resolve problems you might encounter while working with DB2 products. One tutorial introduces you to the DB2 problem determination facilities and tools available, and helps you decide when to use them. Other tutorials deal with related topics, such

as ″Database Engine Problem Determination″, ″Performance Problem Determination″, and ″Application Problem Determination″.

See the full set of DB2 problem determination tutorials on the DB2 Technical Support site at http://www.ibm.com/software/data/support/pdm/db2tutorials.html

Related concepts:

- "DB2 Information Center" on page 406
- "Introduction to problem determination - DB2 Technical Support tutorial" in the *Troubleshooting Guide*

Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully. The following list specifies the major accessibility features in DB2® Version 8 products:

- All DB2 functionality is available using the keyboard for navigation instead of the mouse. For more information, see "Keyboard input and navigation."
- You can customize the size and color of the fonts on DB2 user interfaces. For more information, see "Accessible display."
- DB2 products support accessibility applications that use the Java™ Accessibility API. For more information, see "Compatibility with assistive technologies" on page 428.
- DB2 documentation is provided in an accessible format. For more information, see "Accessible documentation" on page 428.

Keyboard input and navigation

Keyboard input

You can operate the DB2 tools using only the keyboard. You can use keys or key combinations to perform operations that can also be done using a mouse. Standard operating system keystrokes are used for standard operating system operations.

For more information about using keys or key combinations to perform operations, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard navigation

You can navigate the DB2 tools user interface using keys or key combinations.

For more information about using keys or key combinations to navigate the DB2 Tools, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard focus

In UNIX® operating systems, the area of the active window where your keystrokes will have an effect is highlighted.

Accessible display

The DB2 tools have features that improve accessibility for users with low vision or other visual impairments. These accessibility enhancements include support for customizable font properties.

Font settings

You can select the color, size, and font for the text in menus and dialog windows, using the Tools Settings notebook.

For more information about specifying font settings, see Changing the fonts for menus and text: Common GUI help.

Non-dependence on color

You do not need to distinguish between colors in order to use any of the functions in this product.

Compatibility with assistive technologies

The DB2 tools interfaces support the Java Accessibility API, which enables you to use screen readers and other assistive technologies with DB2 products.

Accessible documentation

Documentation for DB2 is provided in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to view documentation according to the display preferences set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen-reader.

Related concepts:

- "Dotted decimal syntax diagrams" on page 428

Related tasks:

- "Keyboard shortcuts and accelerators: Common GUI help"
- "Changing the fonts for menus and text: Common GUI help"

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.
- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one

data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Related concepts:

- "Accessibility" on page 427

Related tasks:

- "Keyboard shortcuts and accelerators: Common GUI help"

Related reference:

- "How to read the syntax diagrams" in the *SQL Reference, Volume 2*

Common Criteria certification of DB2 Universal Database products

DB2 Universal Database is being evaluated for certification under the Common Criteria at evaluation assurance level 4 (EAL4). For more information about Common Criteria, see the Common Criteria web site at: http://niap.nist.gov/cc-scheme/.

Appendix L. 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/region 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/region 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 that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
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 document 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 illustrate 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 are trademarks of International Business Machines Corporation in the United States, other countries, or both, and have been used in at least one of the documents in the DB2 UDB documentation library.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 UDB documentation library:

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

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

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

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

S

T

U

V

W

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-888-426-4343 to learn about available service options
- 1-800-IBM-4YOU (426-4968) for DB2 marketing and sales

In Canada, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-800-465-9600 to learn about available service options
- 1-800-IBM-4YOU (1-800-426-4968) for DB2 marketing and sales

To locate an IBM office in your country or region, check IBM's Directory of Worldwide Contacts on the web at http://www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at http://www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, product downloads, newsgroups, FixPaks, news, and links to web resources.

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) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at www.ibm.com/planetwide

Printed in USA

SC09-4820-01

Spine information:

IBM® DB2 Universal Database™ Administration Guide: Implementation Version 8.2