

DB2®



DB2 Version 9
for Linux, UNIX, and Windows

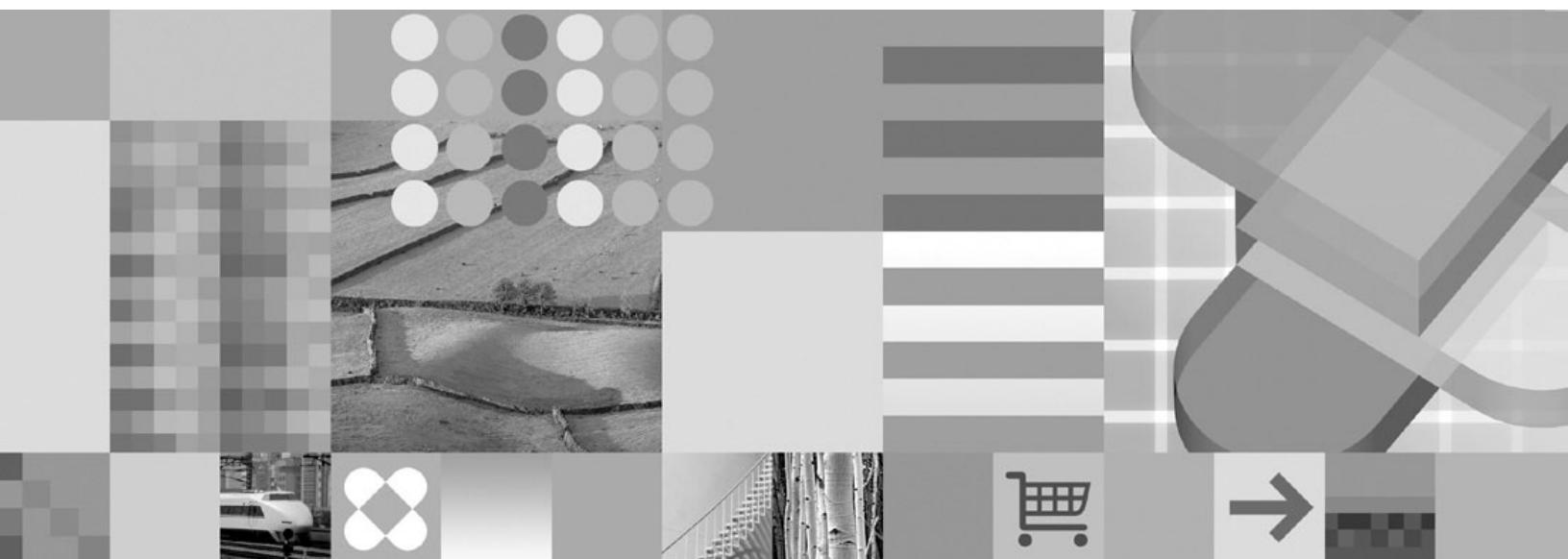


Administration Guide: Implementation

DB2

IBM

DB2 Version 9
for Linux, UNIX, and Windows



Administration Guide: Implementation

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

Edition Notice

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

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.

© **Copyright International Business Machines Corporation 1993, 2006. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book	ix
Who should use this book	x
How this book is structured	x

Part 1. Implementing Your Design . . 1

Chapter 1. Before creating a database. . 3

Working with instances	4
Starting a DB2 instance (Linux, UNIX).	4
Starting a DB2 instance (Windows)	4
Attaching to and detaching from a non-default instance of the database manager	5
Grouping objects by schema	6
Enabling inter-partition query parallelism.	7
Enabling intra-partition parallelism for queries	7
Enabling intra-partition parallelism for utilities	8
Enabling large page support in a 64-bit environment (AIX)	12
Stopping an instance (Linux, UNIX)	13
Stopping an instance (Windows)	14
Working with multiple DB2 copies	15
Multiple DB2 copies roadmap	15
Multiple instances of the database manager	16
Multiple DB2 copies on the same computer (Windows).	17
Changing the Default DB2 copy after installation (Windows).	21
Client connectivity using multiple DB2 copies (Windows).	22
Setting the DAS when running multiple DB2 copies (Windows)	24
Setting the default instance when using multiple DB2 copies (Windows).	25
Managing DB2 copies (Windows)	26
Running multiple instances concurrently (Windows).	27
Removing DB2 copies (Linux, UNIX, and Windows)	28
Working with partitioned databases	29
Management of database server capacity	29
Multiple logical partitions	30
Fast communications manager (FCM) communications	32
Preparing to create a database	33
Designing logical and physical database characteristics	34
Instance creation.	34
Instance management	36
Setting the DB2 environment automatically on UNIX	43
Setting the DB2 environment manually on UNIX	44
Automatic client rerouting	44
Automatic storage	54
License management	64
Registry and environment variables	65

Configuration files and parameters	80
Database history file	87

Chapter 2. Creating and using the DB2 Administration Server (DAS) 91

DB2 Administration Server	91
Creating a DB2 administration server (DAS)	93
Starting and stopping the DB2 administration server (DAS)	94
Listing the DB2 administration server (DAS)	95
Configuring the DB2 administration server (DAS)	95
Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration	96
Notification and contact list setup and configuration	100
DB2 administration server (DAS) Java virtual computer setup.	101
Security considerations for the DB2 administration server (DAS) on Windows	102
Updating the DB2 administration server (DAS) on UNIX	102
Removing the DB2 administration server (DAS)	103
Setting up DB2 administration server (DAS) with Enterprise Server Edition (ESE) systems	104
DB2 administration server (DAS) configuration on Enterprise Server Edition (ESE) systems	106
Discovery of administration servers, instances, and databases.	107
Discovering and hiding server instances and databases.	108
Setting discovery parameters	109
Setting up the DB2 administration server (DAS) to use the Configuration Assistant and the Control Center.	110
Updating a DB2 administration server (DAS) configuration for discovery	110
DB2 administration server (DAS) first failure data capture (FFDC)	111

Chapter 3. Creating a database. 113

Creating a database	113
Initial database partition groups	115
Creating and managing database partitions and database partition groups	115
Creating database partition groups	115
Managing database partitions	116
Adding and dropping database partitions	119
Redistributing data in a database partition group	128
Error recovery when adding database partitions	128
Issuing commands to multiple database partitions.	130
Using Windows database partition servers	143
Creating table spaces	147
Table spaces	148

Defining initial table spaces	148
Creating a table space	149
Automatic resizing of table spaces	154
Creating a system temporary table space	158
Creating a user temporary table space	159
Creating table spaces without file system caching	159
Table spaces in database partition groups	163
Attaching a direct disk access device	163
Setting up a direct disk access device on Linux	164
Creating a buffer pool	166
Creating buffer pools for partitioned databases	167
Creating schemas	168
Creating a schema	168
Setting a schema	169
Copying a schema	170
Restarting a failed copy schema operation	173
System catalog tables	175
Cataloging a database	176
Cataloging database systems	177
Database directories, directory services, and logs	178
Local database directory	178
System database directory	178
Viewing the local or system database directory files	179
Node directory	179
Changing database directory information	180
Updating the directories with information about remote database server computers	180
Lightweight Directory Access Protocol (LDAP) directory service	181
Database recovery log	182
Administration notification log	182
Binding utilities to the database	183
Generating DDL statements for database objects	183
Quiescing and unquiescing databases	186

Chapter 4. Creating tables and other related table objects 187

Space compression for tables	187
Space value compression for new tables	187
Data row compression	188
Table creation	189
Creating a table using the Create Table wizard	190
Creating a table in multiple table spaces	190
Creating a table in a partitioned database environment	191
Creating partitioned tables	193
Details of partitioned tables	194
Approaches to defining ranges on partitioned tables	195
Approaches to migrating existing tables and views to partitioned tables	198
Creating materialized query tables	201
Creating a materialized query table	201
Creating a user-maintained materialized query table	204
Populating a user-maintained materialized query table	205
Partitioned materialized query table behavior	206
Creating a new source table using db2look	210

Creating a staging table	211
Creating a user-defined temporary table	212
Creating range-clustered tables	213
Examples of range-clustered tables	213
How the query compiler works with range-clustered tables	215
Guidelines for using range-clustered tables	216
Creating typed tables	216
Creating a hierarchy table or a typed table	216
Populating a typed table	217
Creating and populating a table	217
Details on creating and populating a table	219
Defining columns	219
Defining keys and constraints	223
Defining dimensions on a table	235
Loading data into a table using the Load wizard	237
Making a table in no data movement mode fully accessible	238
Quiescing tables	239
Defining triggers	240
Creating triggers	240
Trigger dependencies	242
Defining UDFs and UDTs	243
User-defined functions (UDFs) or methods	243
Details on creating a user-defined function (UDF) or method	244
User-defined types (UDTs)	246
Details on creating a user-defined type (UDT)	247
Source data types	249
Length limits for source data types	250
Creating a view	251
Creating an alias	254
Creating indexes	255
Creating an index	256
Index, index extension, or index specification	258
Using an index	260
Options on the CREATE INDEX statement	261
User-defined extended index types	265
Creating user-defined extended index types	266
Showing related objects	270
Validating related objects	271
Estimating space requirements for tables and indexes	272

Chapter 5. Altering a database 275

Altering an instance	275
Changing instances (UNIX only)	275
Details on changing instances	276
Changing node and database configuration files	279
Changing the database configuration across multiple database partitions	281
Altering a database	281
Altering a database partition group	281
Managing database partitions from the Control Center	282
Altering a buffer pool	283
Altering a table space	284
Details on altering a table space	285
Dropping a database	293
Dropping a schema	294

Chapter 6. Altering tables and other related table objects 295

Modifying tables	295
Space value compression for existing tables	295
Copying tables	296
Altering a table.	297
Changing table attributes	298
Changing table properties	299
Altering columns and rows.	300
Altering keys and constraints	309
Changing distribution keys.	318
Altering an identity column	318
Altering a sequence	319
Dropping a sequence.	320
Dropping or removing columns	320
Defining a generated column on an existing table	321
Declaring a table volatile	323
Using a stored procedure to alter a table	324
Modifying indexes	326
Renaming an existing table or index.	326
Dropping an index, index extension, or an index specification.	327
Modifying triggers	328
Updating view contents using triggers	328
Dropping a trigger	329
Modifying aliases and views	330
Altering or dropping a view	330
Recovering inoperative views	331
Dropping aliases	332
Modifying UDFs and UDTs	333
Altering a user-defined structured type.	333
Dropping a user-defined function (UDF), function mapping, or method	333
Dropping a user-defined type (UDT) or type mapping	334
Modifying materialized query tables	335
Altering materialized query table properties	335
Refreshing the data in a materialized query table	336
Modifying partitioned tables	336
Altering partitioned tables	336
Guidelines and restrictions on altering partitioned tables with attached or detached data partitions	338
Rotating data in a partitioned table	339
Examples of rolling in and rolling out partitioned table data.	342
Attaching a data partition	346
Resolving a mismatch when trying to attach a data partition to a partitioned table	348
Detaching a data partition	352
Attributes of detached data partitions	354
Adding data partitions to partitioned tables	356
Dropping a data partition	358
Updating table and view contents using the MERGE statement.	360
Recovering inoperative summary tables	361
Dropping or deleting tables	362
Deleting and updating rows of a typed table	362
Deleting the contents of staging tables	362

Dropping a table	363
Dropping a user-defined temporary table	364
Dropping a materialized query or staging table	365
Statement dependencies when changing objects	366

Chapter 7. Using the DB2 administration tools 369

Starting the server DB2 administration tools	369
Shutting down server DB2 administration tools	369
Finding service level information about the DB2 administration tools environment.	370
Using the DB2 database help	370
Environment-specific information.	371
Menus and toolbars	371
DB2 toolbar	371
DB2 secondary toolbar	373
DB2 Tools menu	374
DB2 Help menu	375
Control Center	376
Control Center overview	376
Control Center Legend	380
Opening new Control Centers.	382
Creating database objects	382
Changing system names displayed in the Control Center	383
Getting help in the Control Center	385
Using advisors, wizards, and launchpads to perform tasks quickly and easily	385
Wizard overviews	387
Control Center object tree and details view	388
Extending the Control Center	394
License Center	411
License Center overview.	411
Adding licenses	412
Changing licenses and policies	413
Viewing licensing information.	413
Viewing license policy information	414
Viewing authorized user infraction information	415
Viewing and resetting compliance details	415
Removing licenses.	416
Task Center and Journal.	416
Task Center overview	416
Journal overview	418
Enabling scheduling settings in the Task Center Scheduler.	419
Scheduler.	420
Success code sets	420
Running tasks immediately.	421
Scheduling a task	422
Changing the default notification message.	423
Creating a database for the DB2 tools catalog	424
Creating or editing a task	425
Selecting users and groups for new tasks	427
Managing contacts	428
Managing saved schedules	429
Managing success code sets	430
Managing task categories	431
Tools Settings	432
Tools Settings overview	432
Setting the server administration tools startup property	434

Setting a command statement termination character	434
Setting up access to DB2 contextual help and documentation	435
Setting startup and default options for the DB2 administration tools	436
Changing the fonts for menus and text	437
Setting DB2 UDB OS/390 and z/OS utility execution options	437
DB2 for z/OS health monitor	441
Enabling or disabling notification using the Health Center Status Beacon	448
Setting the default scheduling scheme	449
Setting Command Editor options	449
Setting IMS options	450
Visual Explain	451
Visual Explain overview	451
Visual Explain concepts	452
Dynamically explaining an SQL or an XQuery statement	464
Creating an access plan using the Command Editor	465
Explain tables	466
Guidelines for creating indexes	467
Out-of-date access plans	467
Retrieving the access plan when using LONGDATACOMPAT	468
Using RUNSTATS	468
Viewing SQL or XQuery statement details and statistics	469
Viewing a graphical representation of an access plan	473
Viewing explainable statements for a package	474
Viewing the history of previously explained query statements	476
Visual Explain support for earlier and later releases	478

Part 2. Database Security 479

Chapter 8. Controlling database access 481

Security issues when installing the DB2 database manager	481
Acquiring Windows users' group information using an access token.	483
Details on security based on operating system	485
Windows platform security considerations for users	485
Windows local system account support.	485
Extended Windows security using DB2ADMNS and DB2USERS groups	486
UNIX platform security considerations for users	489
Location of the instance directory	489
Security plug-ins	490
Authentication methods for your server	490
Authentication considerations for remote clients	495
Partitioned database authentication considerations	496
Kerberos authentication details	496
Authorization, privileges, and object ownership	501

Details on privileges, authorities, and authorization	506
System administration authority (SYSADM)	506
System control authority (SYSCTRL).	507
System maintenance authority (SYSMAINT)	508
Security administration authority (SECADM)	508
Database administration authority (DBADM)	509
System monitor authority (SYSMON)	510
LOAD authority	511
Database authorities	511
Authorization ID privileges.	513
Implicit schema authority (IMPLICIT_SCHEMA) considerations	513
Schema privileges	514
Table space privileges	515
Table and view privileges	515
Package privileges.	517
Index privileges	518
Sequence privileges	518
Routine privileges	518
Controlling access to database objects	519
Details on controlling access to database objects	519
Granting privileges	519
Revoking privileges	521
Managing implicit authorizations by creating and dropping objects	522
Establishing ownership of a package	523
Indirect privileges through a package	523
Indirect privileges through a package containing nicknames	524
Controlling access to data with views	525
Monitoring access to data using the audit facility.	527
Data encryption	527
Granting database authorities to new groups	529
Granting database authorities to new users	529
Granting privileges to new groups	530
Granting privileges to new users	534
Label-based access control (LBAC)	538
Label-based access control (LBAC) overview	538
LBAC security policies	540
LBAC security label components	541
LBAC security labels	547
Format for security label values	549
How LBAC security labels are compared	550
LBAC rule sets	551
LBAC rule exemptions	556
Built-in functions for dealing with LBAC security labels	557
Protection of data using LBAC	558
Reading of LBAC protected data	560
Inserting of LBAC protected data.	563
Updating of LBAC protected data	565
Deleting or dropping of LBAC protected data	569
Removal of LBAC protection from data	572
Lightweight directory access protocol (LDAP) directory services	573
Lightweight Directory Access Protocol (LDAP) overview	573
Supported LDAP client and server configurations	575
Support for Active Directory	575

Configuring DB2 to use Active Directory	576
Configuring DB2 in the IBM LDAP environment	576
Creating an LDAP user	577
Configuring the LDAP user for DB2 applications	578
Registration of DB2 servers after installation	578
Update the protocol information for the DB2 server	580
Rerouting LDAP clients to another server	580
Catalog a node alias for ATTACH	581
Deregistering the DB2 server	582
Registration of databases in the LDAP directory	582
Attaching to a remote server in the LDAP environment.	583
Deregistering the database from the LDAP directory	584
Refreshing LDAP entries in local database and node directories	584
Searching the LDAP servers	585
Registering host databases in LDAP	586
Setting DB2 registry variables at the user level in the LDAP environment	587
Enabling LDAP support after installation is complete	588
Disabling LDAP support	589
LDAP support and DB2 Connect	589
Security considerations in an LDAP environment.	589
Security considerations for Active Directory	590
Extending the LDAP directory schema with DB2 object classes and attributes	591
Extending the directory schema for Active Directory	591
DB2 objects in the Active Directory	593
Netscape LDAP directory support and attribute definitions	593
Extending the directory schema for IBM Tivoli Directory Server	595
Extending the directory schema for Sun One Directory Server	596
LDAP object classes and attributes used by DB2	598
Tasks and required authorizations	608
Using the system catalog for security issues	609
Details on using the system catalog for security issues	610
Retrieving authorization names with granted privileges.	610
Retrieving all names with DBADM authority	611
Retrieving names authorized to access a table	612
Retrieving all privileges granted to users	613
Securing the system catalog view.	613
Security considerations	616
Introduction to firewall support	619
Screening router firewalls	619
Application proxy firewalls.	620
Circuit level firewalls.	620
Stateful multi-layer inspection (SMLI) firewalls	620
Chapter 9. Auditing DB2 database activities	621
Introduction to the DB2 database audit facility	621

Audit facility behavior	623
Audit facility usage	624
Working with DB2 audit data in DB2 tables	628
Working with DB2 audit data in DB2 tables	628
Creating tables to hold the DB2 audit data	628
Creating DB2 audit data files	631
Loading DB2 audit data into tables	632
Selecting DB2 audit data from tables	635
Audit facility messages	636
Audit facility record layouts (introduction)	636
Details on audit facility record layouts	637
Audit record layout for AUDIT events	637
Audit record layout for CHECKING events	638
Audit record object types	639
List of possible CHECKING access approval reasons	640
List of possible CHECKING access attempted types	641
Audit record layout for OBJMAINT events	643
Audit record layout for SECMAINT events	645
List of possible SECMAINT privileges or authorities	647
Audit record layout for SYSADMIN events	650
List of possible SYSADMIN audit events	650
Audit record layout for VALIDATE events	651
Audit record layout for CONTEXT events	652
List of possible CONTEXT audit events	653
Audit facility tips and techniques.	654
Controlling DB2 database audit facility activities	655

Part 3. Appendixes 661

Appendix A. Conforming to the naming rules 663

General naming rules.	663
DB2 database object naming rules	663
Delimited identifiers and object names	665
User, user ID and group naming rules	666
Federated database object naming rules	666
Additional restrictions and recommendations regarding the use of schema names	667
Maintaining passwords on servers	667
Workstation naming rules	667
Naming rules in an NLS environment	668
Naming rules in a Unicode environment	669

Appendix B. Using Windows Management Instrumentation (WMI) support 671

Introduction to Windows Management Instrumentation (WMI)	671
DB2 database system integration with Windows Management Instrumentation	672

Appendix C. Using Windows security 675

DB2 and Windows security introduction	675
A scenario with server authentication (Windows)	676
A scenario with client authentication and a Windows client machine.	677

Support for global groups (on Windows)	677
Using a backup domain controller with DB2 database systems	677
User authentication with DB2 for Windows	678
User name and group name restrictions (Windows)	678
Groups and user authentication on Windows	679
Trust relationships between domains on Windows	679
DB2 database system and Windows security service.	680
Installing DB2 on a backup domain controller	680
Authentication with groups and domain security (Windows)	681
Authentication using an ordered domain list	682
Domain security support (Windows)	683

Appendix D. Using the Windows Performance Monitor 685

Windows performance monitor introduction	685
Registering DB2 with the Windows performance monitor	685
Enabling remote access to DB2 performance information	686
Displaying DB2 database and DB2 Connect performance values	687
Windows performance objects	687
Accessing remote DB2 database performance information	688

Resetting DB2 performance values	688
--	-----

Appendix E. DB2 Database technical information 691

Overview of the DB2 technical information	691
Documentation feedback	691
DB2 technical library in PDF format.	692
Ordering printed DB2 books	694
Displaying SQL state help from the command line processor.	695
Accessing different versions of the DB2 Information Center	695
Displaying topics in your preferred language in the DB2 Information Center	696
Updating the DB2 Information Center installed on your computer or intranet server.	697
DB2 Visual Explain tutorial.	698
DB2 troubleshooting information.	699
Terms and Conditions	699

Appendix F. Notices 701

Trademarks	703
----------------------	-----

Index 705

Contacting IBM 719

About this book

The Administration Guide in its two volumes provides information necessary to use and administer the DB2[®] relational database management system (RDBMS) products, and includes:

- Information about database planning and design (found in *Administration Guide: Planning*)
- Information about implementing and managing databases (found in *Administration Guide: Implementation*)

In Version 9, the information about configuring and tuning your database environment to improve performance can be found in the *Performance Guide*.

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 command-line interface. From this interface, you can also execute SQL and XQuery 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 manage and administer your data and database components. You can invoke the Control Center using the db2cc command on a Linux[®] or Windows[®] command line, or using the Start menu on Windows platforms. The Control Center presents your database components as a hierarchy of objects in an object tree, which includes your systems, instances, databases, tables, views, triggers, and indexes. From the tree you can perform actions on your database objects, such as creating new tables, reorganizing data, configuring and tuning databases, and backing up and restoring databases, database partitions, and table spaces. In many cases, wizards and launchpads are available to help you perform these tasks more quickly and easily.

The Control Center is available in three views:

- *Basic*. This view provides you with the core DB2 functions. From this view you can work with all the databases to which you have been granted access, including their related objects such as tables and stored procedures. It provides you with the essentials for working with your data.
- *Advanced*. This view provides you with all of the objects and actions available in the Control Center. Use this view if you are working in an enterprise environment and you want to connect to DB2 UDB Version 8 for z/OS or DB2 Version 9 for z/OS (DB2 for z/OS[®]) or IMS[™].
- *Custom*. This view provides you with the ability to tailor the Control Center to your needs. You select the objects and actions that you want to appear in your view.

For help on using the Control Center, select **Getting started** from the **Help** pull-down on the Control Center window.

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

- The **Activity Monitor** helps you monitor application performance and concurrency, resource consumption, and SQL statement usage of a database or database partition.
- The **Command Editor** is used to generate, edit, run, and manipulate SQL and XQuery statements; IMS and DB2 commands; work with the resulting output; and to view a graphical representation of the access plan for explained SQL and XQuery statements.
- The **Configuration Assistant** is used to configure and maintain the database objects that your applications will be using.
- The **Health Center** helps you resolve performance and resource allocation problems.
- The **Indoubt Transaction Manager** is used to display indoubt transactions, that is, the transactions that are waiting to be committed, rolled back, or forgotten for a selected database and one or more selected partitions.
- The **License Center** is used to display license status and usage information for DB2 products installed on your system. You can also use the License Center to configure your system for license monitoring.
- The **Task Center** is used to schedule jobs that are to run unattended. The Journal can be used to view historical information about tasks, database actions and operations, messages, and notifications.
- The **Memory Visualizer** helps you monitor the memory-related performance of an instance and all of its databases organized in a hierarchical tree.
- **Tools Settings** is used to change the settings for the Control Center, Health Center, and the Information Center.
- **Visual Explain** is used to display access plan graphs for explained SQL or XQuery statements. You can use the information in the graph to tune your queries.

For more information about the Control Center and the administration tools listed above, refer to Chapter 7, or search for them in the DB2 Information Center.

Who should use this book

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

How this book is structured

This book contains information about the following major topics:

Implementing Your Design

- Chapter 1, Chapter 1, “Before creating a database,” describes the prerequisites needed before creating a database and the objects within a database.
- Chapter 2, 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, Chapter 3, “Creating a database,” describes the tasks associated with creating a database and the objects within a database.

- Chapter 4, Chapter 4, “Creating tables and other related table objects,” describes how to create tables with specific characteristics when implementing your database design.
- Chapter 5, 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, 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.
- Chapter 7, Chapter 7, “Using the DB2 administration tools,” describes the graphical user interface tools and includes some tasks that can only be performed using the graphical user interface. This chapter also discusses 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.

Database Security

- Chapter 8, Chapter 8, “Controlling database access,” describes how you can control access to your database’s resources.
- Chapter 9, Chapter 9, “Auditing DB2 database 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 Windows Management Instrumentation (WMI) support,” provides information about how DB2 can be managed using Windows Management Instrumentation (WMI).
- Appendix C, “Using Windows security,” describes how DB2 works with Windows security.
- Appendix D, “Using the Windows Performance Monitor,” describes how to use the Windows Performance Monitor to collect DB2 performance 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 and XQuery statements in the command line processor.
- Through SQL and XQuery statements in applications using application programming interfaces (APIs).
- Through the Control Center.

In this and other chapters, the Control Center method for completing tasks is highlighted by placing it within a box. This is followed immediately by a comparable method using the command line, and if applicable, using an API. In some cases, there may be tasks showing only one method. When working with the Control Center, recall that you can use the help to obtain more detail than the overview information found in this manual.

For information on SQL and XQuery statements, refer to the *SQL Reference* manual. For information on command line processor commands, refer to the *Command Reference* manual. For information on APIs, refer to the *Administrative API Reference* manual. For information on the Control Center and other administration tools, refer to Chapter 7.

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. Chapter 6 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 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 Database. Refer to the *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.

References to Windows will mean all supported Windows operating systems.

Working with instances

Before you implement a database, you should understand the prerequisite concepts and tasks described in this section.

Starting a DB2 instance (Linux, UNIX)

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

- Connecting to a database on the instance
- Precompiling an application
- Binding a package to a database
- Accessing host databases.

Prerequisites:

Before you start a DB2 instance on your system:

1. Log in with a user ID or name that has SYSADM, SYSCTRL, or SYSMANT 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:

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.

To start the instance using the command line, enter:

```
db2start
```

Note: When you run commands to start or stop an instance's database manager, the DB2 database manager applies the command to the current instance. For more information, see [Setting the current instance environment variables](#).

Related tasks:

- "Setting the current instance environment variables" on page 67
- "Starting a DB2 instance (Windows)" on page 4
- "Stopping an instance (Linux, UNIX)" on page 13

Starting a DB2 instance (Windows)

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

- Connecting to a database on the instance

- Precompiling an application
- Binding a package to a database
- Accessing host databases.

Prerequisites:

In order to successfully launch the DB2 database instance as a service from **db2start**, the user account must have the correct privilege as defined by the Windows operating system to start a Windows service. The user account can be a member of the Administrators, Server Operators, or Power Users group. When extended security is enabled, only members of the DB2ADMNS and Administrators groups can start the database by default.

Procedure:

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.

To start the instance using the command line, enter:

```
db2start
```

Note: When you run commands to start or stop an instance’s database manager, the DB2 database manager applies the command to the current instance. For more information, see Setting the current instance environment variables.

The **db2start** command will launch the DB2 database instance as a Windows service. The DB2 database instance on Windows can still be run as a process by specifying the **/D** switch when invoking **db2start**. The DB2 database instance 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 a DB2 instance as a process in a partitioned database environment.

Related tasks:

- “Setting the current instance environment variables” on page 67
- “Starting a DB2 instance (Linux, UNIX)” on page 4
- “Stopping an instance (Windows)” on page 14
- “Stopping an instance (Linux, UNIX)” on page 13

Attaching to and detaching from a non-default instance of the database manager

To attach to another instance of the database manager, which might 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
```

To attach to an instance from a client application, call the sqleatin API.

After performing maintenance activities for the testdb2 instance, you can then **DETACH** from that instance by running the following command:

```
db2 detach
```

To detach from an instance from a client application, call the sqledtin API.

Related reference:

- “ATTACH command” in *Command Reference*
- “DETACH command” in *Command Reference*

Grouping objects by schema

Database object names might be made up of a single identifier or they might 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 want 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 and XQuery 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 and XQuery 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:

- “System catalog tables” on page 175

Related tasks:

- “Creating a schema” on page 168

Related reference:

- “CURRENT SCHEMA special register” in *SQL Reference, Volume 1*
- “SET SCHEMA statement” in *SQL Reference, Volume 2*

Enabling inter-partition query parallelism

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

Note: 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.

Related concepts:

- “Partitioned database environments” in *Administration Guide: Planning*
- “Database partition group design” in *Administration Guide: Planning*
- “Database partition and processor environments” in *Administration Guide: Planning*
- “Adding database partitions in a partitioned database environment” on page 123

Related tasks:

- “Redistributing data across database partitions” in *Performance Guide*
- “Enabling database partitioning in a database” on page 9
- “Enabling intra-partition parallelism for queries” on page 7

Enabling intra-partition parallelism for queries

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 database 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 that 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 *Performance Guide*
- “Parallel processing information” in *Performance Guide*

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “dft_degree - Default degree configuration parameter” in *Performance Guide*
- “intra_parallel - Enable intra-partition parallelism configuration parameter” in *Performance Guide*
- “max_querydegree - Maximum query degree of parallelism configuration parameter” in *Performance Guide*
- “BIND command” in *Command Reference*
- “PRECOMPILE command” in *Command Reference*
- “CURRENT DEGREE special register” in *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 database partitioning in a database

The decision to create a multi-partition database 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 database partitioning can offer.

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

When running in a partitioned database environment, you can create a database from any database partition that exists in the `db2nodes.cfg` file using the **CREATE DATABASE** command or the `sqlcrea()` application programming interface (API).

Before creating a multi-partition database, you must select which database partition will be the catalog partition for the database. You can then create the database directly from that database partition, or from a remote client that is attached to that database partition. The database partition to which you attach and execute the **CREATE DATABASE** command becomes the *catalog partition* for that particular database.

The catalog partition 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 (for example, wrappers, servers, and nicknames) are stored in the system catalog tables at this database partition.

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 partitions 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 partition 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 `sqlbdbir` and is located in the `sqllib` directory under your home directory, or under the directory where DB2 database 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 environment. When working on Windows, the system database directory is located in the instance directory.

Also resident in the `sqlbdir` directory is the system intention file. It is called `sqlbins`, 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 database partitions making up the database.

Configuration parameters have to be modified to take advantage of database 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 environment include *conn_elapse*, *fcu_num_buffers*, *fcu_num_channels*, *max_connretries*, *max_coordagents*, *max_time_diff*, *num_poolagents*, and *stop_start_time*.

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “CREATE DATABASE command” in *Command Reference*
- “sqlcrea API - Create database” in *Administrative API Reference*

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 database partitions. Inter-partition parallelism for data loading can be overridden by specifying `OUTPUT_DBPARTNUMBS`. The load utility also intelligently enables database partitioning parallelism depending on the size of the target database partitions. `MAX_NUM_PART_AGENTS` can be used to control the maximum degree of parallelism selected by the load utility. Database partitioning parallelism can be overridden by specifying `PARTITIONING_DBPARTNUMS` when `ANYORDER` is also specified.

Related concepts:

- “Load overview” in *Data Movement Utilities Guide and Reference*
- “Load in a partitioned database environment - overview” in *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 computer.

Related reference:

- “CREATE INDEX statement” in *SQL Reference, Volume 2*
- “intra_parallel - Enable intra-partition parallelism configuration parameter” in *Performance Guide*

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 *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 *Command Reference*

Enabling large page support in a 64-bit environment (AIX)

In addition to the traditional page size of 4 KB, the POWER4™ processor in the IBM® eServer™ pSeries® systems also supports a new 16 MB page size. AIX 5L™ for POWER™ Version 5.1 with the 5100-02 Recommended Maintenance package, or Version 5.2, contain support for pages with a 16 MB size. When running under this environment, IBM DB2 Version 9.1 for AIX® 64-bit Edition can be enabled to use these large pages.

Large page usage is primarily intended to provide performance improvements to high performance computing applications. Applications that require intensive memory access and that use large amounts of virtual memory may obtain performance improvements by using large pages.

Notes:

1. For detail instructions on how to run the **vmtune** or the **vmo** command, refer to your AIX manuals.
2. You should be extremely cautious when configuring your system for pinning memory and supporting large pages. Pinning too much memory results in heavy paging activities for the memory pages that are not pinned. Allocating too much physical memory to large pages will degrade system performance if there is insufficient memory to support the 4 KB pages.
3. Setting the DB2_LGPAGE_BP registry variable also implies that the memory is pinned.

Prerequisites:

You are working in an AIX 5.x or later 64-bit environment. You must have root authority to work with the AIX operating system commands.

Procedure:

To enable large page support, you must:

1. Configure your AIX server for large page support:

For AIX 5.1 operating systems: Issue the **vmtune** command with the following flags:

```
vmtune -g <LargePageSize> -L <LargePages>
```

For AIX 5.2 operating systems: Issue the **vmo** command with the following flags:

```
vmo -r -o lpgg_size=<LargePageSize> lpgg_regions=<LargePages>
```

where

<LargePageSize>

Specifies the size in bytes of the hardware-supported large pages.

<LargePages>

Specifies the number of large pages to reserve.

For example, if you need to allocate 25 GB for large page support, run the command as follows:

For AIX 5.1 operating systems:

```
vmtune -g 16777216 -L 1600
```

On AIX 5.2 operating systems:

```
vmo -r -o lpgg_size=16777216 lpgg_regions=1600
```

2. Run the **bosboot** command so that the previously run **vmtune** command or **vmo** command will take effect following the next system boot.

3. After the server comes up, enable it for pinned memory:

For AIX 5.1 operating systems: Issue the **vmtune** command with the following flags:

```
vmtune -S 1
```

For AIX 5.2 operating systems: Issue the **vmo** command with the following flags:

```
vmo -o v_pinshm=1
```

4. Use the **db2set** command to set the DB2_LGPAGE_BP registry variable to "YES", then start DB2:

```
db2set DB2_LGPAGE_BP=YES  
db2start
```

Related concepts:

- "Database managed space" in *Administration Guide: Planning*
- "System managed space" in *Administration Guide: Planning*
- "Table space design" in *Administration Guide: Planning*

Stopping an instance (Linux, UNIX)

You might 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 SYSMANT 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 SYSMANT 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:

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**.

To stop the instance using the command line, enter:

```
db2stop
```

You can use the `db2stop` command to stop, or drop, individual database partitions within a partitioned database environment. When working in a partitioned database environment 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.

Note: When you run commands to start or stop an instance's database manager, the DB2 database manager applies the command to the current instance. For more information, see *Setting the current instance environment variables*.

Related tasks:

- "Setting the current instance environment variables" on page 67

Related reference:

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

Stopping an instance (Windows)

You might 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 database 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 the DB2 database service 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

- | |
|---|
| <ol style="list-style-type: none">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 the DB2 database manager in a partitioned database environment, each database partition server is started as a service. Each service must be stopped.

Note: When you run commands to start or stop an instance’s database manager, the DB2 database manager applies the command to the current instance. For more information, see *Setting the current instance environment variables*.

Related tasks:

- “Setting the current instance environment variables” on page 67

Related reference:

- “db2stop - Stop DB2 command” in *Command Reference*

Working with multiple DB2 copies

This section describes how to run and administer multiple DB2 copies on the same computer, including migration, installation, and configuring information. A DB2 Copy refers to one or more installations of DB2 database products in a particular location on the same computer. Each DB2 copy can be at the same or different code levels.

Multiple DB2 copies roadmap

With DB2 Version 9, you can install and run multiple DB2 copies on the same computer. A DB2 Copy refers to one or more installations of DB2 database products in a particular location on the same computer. Each DB2 copy can be at the same or different code levels. The benefits of doing this include:

- The ability to run applications that require different DB2 versions on the same computer at the same time.
- The ability to run independent copies of DB2 products for different functions.
- The ability to test on the same computer before moving the production database to the latter version of the DB2 product.

- For independent software vendors, the ability to embed a DB2 server product into your product and hide the DB2 database from your users. For COM+ applications, we recommend that you use and distribute the *IBM DB2 Driver for ODBC and CLI* with your application instead of the *DB2 Runtime Client* as only one *DB2 Runtime Client* can be used for COM+ applications at a time. The *IBM DB2 Driver for ODBC and CLI* does not have this restriction.

Table 1 lists the relevant topics in each category.

Table 1. Roadmap to multiple DB2 copies information

Category	Related topics
General information and restrictions	<ul style="list-style-type: none"> • Multiple DB2 copies on the same computer (Linux and UNIX) • Multiple DB2 copies on the same computer (Windows)
Migration	<ul style="list-style-type: none"> • Migrating from a system with multiple DB2 copies (Linux and UNIX) • Migrating a DB2 server (Windows) • Migrating DB2 32-bit servers to 64-bit systems (Windows)
Installation	<ul style="list-style-type: none"> • Installing DB2 servers (Linux and UNIX) • Installing DB2 servers (Windows)
Configuration	<ul style="list-style-type: none"> • Changing the Default DB2 copy after installation (Windows) • Client connectivity using multiple DB2 copies (Windows) • Selecting a different DB2 copy for your Windows CLI application • Setting the DAS when running multiple DB2 copies (Windows) • Setting the default instance when using multiple DB2 copies (Windows)
Administration	<ul style="list-style-type: none"> • Listing DB2 products installed on your system (Linux and UNIX) • Managing DB2 copies (Windows) • Running multiple instances concurrently (Windows)
Uninstalling	<ul style="list-style-type: none"> • Removing DB2 copies (Linux, UNIX, and Windows) • Removing DB2 products using the <code>db2_deinstall</code> or <code>doce_deinstall</code> command (Linux and UNIX)

Multiple instances of the database manager

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

You might want to have multiple instances to create the following environments:

- Separate your development environment from your production environment.
- Separately tune each environment for the specific applications it will service.
- Protect sensitive information from administrators. For example, you might want 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 file system. Errors will be returned when the home file system is shared.

DB2 database program files are physically stored in one location on a particular computer. Each instance that is created points back to this location so that 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 Linux or UNIX operating system” on page 36
- “Multiple instances on a Windows operating system” on page 37

Related tasks:

- “Creating additional instances” on page 38

Related reference:

- “ATTACH command” in *Command Reference*
- “Multiple DB2 copies roadmap” on page 15

Multiple DB2 copies on the same computer (Windows)

With DB2 Version 9, you can use multiple DB2 copies on the same computer. Each DB2 copy can be at the same or different code levels. The benefits of doing this include:

- The ability to run applications that require different DB2 versions on the same machine at the same time.
- The ability to run independent copies of DB2 products for different functions.
- The ability to test on the same computer before moving the production database to the latter version of the DB2 product.
- For independent software vendors, the ability to embed a DB2 server product into your product and hide the DB2 database from your users.

A DB2 copy can contain one or more different DB2 products. This refers to the group of DB2 products that are installed at the same location.

Differences when only one DB2 copy is installed:

- During installation, a unique Default DB2 copy name is generated, which you can later change.
- Applications use the Default DB2 copy in an environment similar to the DB2 Version 8 environment.

Differences when multiple DB2 copies are installed on the same computer:

- DB2 Version 8 can coexist with DB2 Version 9, with restrictions described below.

- Optional: You can configure each DB2 copy to use a different Information Center.

Note: You can have only one copy of the DB2 Information Center installed on the same system at the same Release level. Specifically, you can have a Version 8 Information Center and a V9 Information Center, but you cannot have one Information Center at Version 9 FixPak1 and another at Version 9 fix pack 2 on the same machine. You can however configure the DB2 database server to access these Information Centers remotely.

- Only the IBM DB2 .NET Data Provider from the Default copy is registered in the Global Assembly Cache. If Version 8 is installed with Version 9, the IBM DB2 .NET 2.0 Provider from Version 9 is also registered in the Global Assembly Cache. Version 8 does not have a 2.0 .NET provider.
- Each DB2 copy must have unique instance names. For a silent install with NO_CONFIG=YES, the default instance will not be created. However, when you create the instance after the installation, it must be unique. The name of the default instance will be the <DB2 copy Name>, if it is less than 8 characters. If it is more than 8 characters, or if an instance of the same name already exists, a unique name for the instance is generated to ensure uniqueness. This is done by replacing any characters that are not valid for the instance name with underscores and generating the last 2 characters. For performance reasons, the DB2 Control Center should only be used from one DB2 Copy at a single time on a machine.

Restrictions:

For Microsoft COM+ applications, it is recommended that you use and distribute the *IBM DB2 Driver for ODBC and CLI* with your application instead of the *DB2 Runtime Client* as only one *DB2 Runtime Client* can be used for COM+ applications at a time. The *IBM DB2 Driver for ODBC and CLI* does not have this restriction.

Microsoft COM+ applications accessing DB2 data sources are only supported with the default DB2 copy. Concurrent support of COM+ applications accessing different DB2 copies is not supported. If you have DB2 UDB Version 8 installed, you can only use DB2 UDB Version 8 to run these applications. If you have DB2 Version 9 or higher installed, you can change the default DB2 copy using the Default DB2 Copy Selection Wizard, but you can't use them concurrently.

Version 8 coexistence

DB2 Version 8 and DB2 Version 9 can coexist with the restriction that DB2 Version 8 is set as the Default DB2 copy. This cannot be changed unless you uninstall Version 8.

On the server, there can be only one DAS version and it administers instances as follows:

- If the DAS is on Version 9, then it can administer Version 8 and Version 9 instances.
- If the DAS is on Version 8, then it can administer only Version 8 instances. You can migrate your Version 8 DAS, or drop it and create a new Version 9 DAS to administer the Version 8 and Version 9 instances. This is required only if you want to use the Control Center to administer the instances.

Version 8 and Version 9 coexistence and the DB2 .NET Data Provider

In DB2 Version 9, the DB2 .NET Data Provider has System.Transaction support however, this support is only available for the default DB2 copy.

You cannot use the DB2 Version 8 .NET Data Provider if the DB2 Version 9 .NET Data Provider is installed. If Version 8 is installed, the 1.1 .NET Data Provider that is registered in the Global Assembly Cache will be from V8. The 2.0 provider that is registered will be from Version 9.

3rd party applications that run as a service

By default, 3rd party applications that dynamically bind DB2 DLLs, for example, that are linked with db2api.lib, will find the DB2 DLLs in the current PATH. This means that existing applications that are not enabled for multi-version support will use the Default DB2 copy. To work around this, the application can use the db2SelectDB2Copy API prior to loading any DB2 libraries. For more information, see the *Call Level Interface Guide and Reference, Volume 1*.

32- and 64-bit versions on Win x64

DB2 does not support multiple DB2 32- and 64-bit versions installed on Windows. If you install the DB2 64-bit version, the 32-bit version will be removed from the system. This is because the DB2 32- and 64-bit registries reside in different locations.

LDAP and CLI configuration

With DB2 Version 8, if an application needs different LDAP settings, it needs to use a different LDAP user. Otherwise, the CLI configuration will affect all DB2 copies that the LDAP user might potentially use.

Performance counters

Performance counters can be registered for only one DB2 copy at a time and they can monitor only the instances in the DB2 copy in which they are registered. When you switch the Default DB2 copy, the DB2 Selection Wizard de-registers and reregisters the performance counters so that they are active for the Default DB2 copy.

Windows Management Instrumentation (WMI)

Only one version of the WMI provider can be registered at any given time.

Client Connectivity

You can use only one DB2 copy in the same process. For more information, see Client connectivity using multiple DB2 copies (Windows).

Applications that dynamically link DB2 DLLs

Applications that link to DB2 DLLs directly or that use LoadLibrary instead of LoadLibraryEx with the LOAD_WITH_ALTERED_SEARCH_PATH parameter will need to ensure that the initial dependent library is loaded properly. You can use your own coding technique to do this, or you can call the db2envvar.bat file to setup the environment before running the application, or you can call the **db2SelectDB2Copy API**, which can be statically linked into the application.

Visual Studio 2003 plugins:

There can be only one version of the plugins registered on the same computer at the same time. The version of the plugins that is active will be the version that is shipped with the Default DB2 copy.

Licensing:

Licenses need to be registered for each DB2 copy. They are not system-wide. This allows different licenses for different paths and provides the ability for both restricted versions of DB2 copies of the product and full versions of DB2 copies on the same machine.

NT Services:

DB2 NT services will use the <servicename_installationname>. For example, DB2NETSECSERVER_MYCOPY1. The display name also contains the Copy Name appended to it in brackets, for example, DB2 Security Server (MYCOPY1). Instances also include the DB2-<DB2 Copy Name>-<Instance Name>-<Node Number> in the display name, which is shown in the services control panel applet. The actual service name remains as is.

API to select the DB2 copy to use:

You can use the **db2SelectDB2Copy** API to select the DB2 copy that you want your application to use. This API does not require any DLLs. It is statically linked into your application. You can delay the loading of DB2 libraries and call this API first before calling any other DB2 APIs. Note that the function cannot be called more than once for any given process; that is, you cannot switch a process from one DB2 copy to another.

The **db2SelectDB2Copy** API sets the environment required by your application to use the DB2 copy name or the location specified. If your environment is already set up for the copy of DB2 that you want to use, then you do not need to call this API. If, however, you need to use a different DB2 copy, you must call this API before loading any DB2 DLLs within your process. This call can be made only once per process.

Database Partitioning with multiple physical nodes:

Each physical partition must use the same DB2 copy name on all computers.

Using MSCS and Multiple DB2 Copies:

Each DB2 resource must be configured to run in a separate resource monitor.

Related concepts:

- “DB2 .NET Data Provider” in *Developing ADO.NET and OLE DB Applications*
- “What’s new for V9.1: Client and connectivity enhancements summary” in *What’s New*
- “Introduction to Windows Management Instrumentation (WMI)” on page 671

Related tasks:

- “Creating a DB2 administration server (DAS)” on page 93
- “Changing the Default DB2 copy after installation (Windows)” on page 21
- “Configuring the DB2 administration server (DAS)” on page 95
- “Setting the DAS when running multiple DB2 copies (Windows)” on page 24
- “Migrating the DB2 Administration Server (DAS)” in *Migration Guide*

Related reference:

- “dasupdt - Update DAS command” in *Command Reference*

- “db2perfi - Performance counters registration utility command” in *Command Reference*
- “Multiple DB2 copies roadmap” on page 15
- “db2SelectDB2Copy API - Select the DB2 copy to be used by your application” in *Administrative API Reference*

Changing the Default DB2 copy after installation (Windows)

After you have installed DB2 Version 9.1 in several locations on the same computer, you might want to make a different DB2 copy the default copy. The Default DB2 copy is the DB2 copy that is used by applications that access DB2 database products through the default interface. This environment is similar to previous versions of DB2. If you have DB2 Version 8 installed, you need to uninstall or migrate it to Version 9.1 before you can change the Default DB2 copy (on Version 9.1).

Prerequisites:

Multiple DB2 copies (Version 9 or later) are installed on the same computer.

Restrictions:

All DB2 copies are Version 9 or later.

Version 8 and Version 9 DB2 copies can coexist on the same machine, however Version 8 must be the default copy. You cannot change the Version 8 default copy, nor can you run the Default Copy Switcher command, **db2swtch**, unless you uninstall Version 8. If you run the **db2swtch** command when Version 8 exists on the system, you will get a message indicating that you cannot change the default DB2 copy because Version 8 is found on the system.

However, you can work with the Version 9 copy by either running the **db2envar.bat** command or by opening the command window from the Start menu for the copy that you want to work with.

Procedure:

To change the Default DB2 copy using the Default DB2 Selection wizard:

1. Open the Default DB2 Selection wizard: From the Start Menu, select **Programs->IBM DB2-><DB2 copy name>->Default Copy Switcher**. The Default DB2 Selection wizard opens.
2. On the Default DB2 Copy page, select the copy that you want to make the default so that it is highlighted and click **Next** to make it the default copy.
3. On the summary page, the wizard indicates the result of the operation.
4. Invoke the **dasupdt - Update DAS** command to move the DB2 Administration Server (DAS) to the new default copy.

This procedure switches the current Default DB2 copy to the selected DB2 copy and makes the necessary changes to the registry. To access and use the new Default DB2 copy, after you have moved the DAS to the new default copy, open a new command window. You can still access the original Default DB2 copy by using the shortcuts in the Start menu for the original Default DB2 copy.

To change the Default DB2 copy using the command line, invoke the **db2swtch -d <new default copy name>** command.

This procedure unregisters the current Default DB2 copy and registers the specified DB2 copy as the default copy. It also makes the necessary changes to the registry, to the environment variables, to the ODBC and OLE DB drivers, to the WMI registration, and to various other objects, and moves the DAS to specified Default DB2 copy. To access and use the new Default DB2 copy, open a new command window.

The **db2swtch** command can be run from any DB2 copy, Version 9 or greater. For more information on this command, see **db2swtch - Switch default DB2 copy command**.

Related concepts:

- “Multiple DB2 copies on the same computer (Windows)” on page 17

Related tasks:

- “Setting the DAS when running multiple DB2 copies (Windows)” on page 24
- “Removing DB2 copies (Linux, UNIX, and Windows)” on page 28
- “Migrating a DB2 server (Windows)” in *Migration Guide*

Related reference:

- “Multiple DB2 copies roadmap” on page 15
- “dasmigr - Migrate the DB2 administration server command” in *Command Reference*
- “dasupdt - Update DAS command” in *Command Reference*
- “db2envar.bat command” in *Command Reference*
- “db2swtch - Switch default DB2 copy command” in *Command Reference*

Client connectivity using multiple DB2 copies (Windows)

Applications access DB2 databases in several ways. When using multiple DB2 copies, various options are available. Existing applications will continue to function properly.

Note: Only one copy of DB2 can be used within the same process for each of the following modes of connecting to databases.

Restrictions:

See Multiple DB2 copies on the same computer (Windows).

Procedure:

OLE DB

To use a DB2 copy other than the default, in the connection string, specify the IBMDADB driver name for this DB2 copy, which will be of the format: IBMDADB2.\$DB2_COPY_NAME. Some applications might not have the ability to change the connection strings without recompiling, therefore these applications will only work with the Default DB2 copy. If an application uses the default program id, ibmdadb2, or the default clsid, it will always use the Default DB2 copy.

Specifically, you will need to change the value of "provider=IBMDADB2" in the connection string. For example, if the DB2 copy that you want to use is called MY_COPY, you would specify "provider=IBMDADB2.MY_COPY" in the connection string. In case you need to explicitly specify a GUID during installation, a response file keyword, OLEDB_GUID, is used to do this and allows you to enter your own GUID. Otherwise, the generated ID is used, as listed in the DB2 installation log.

Note: If you continue to use the IBMDADB2 provider name, then you will only be able to access data sources from the default DB2 copy.

ODBC

The ODBC driver contains the **DB2 copy Name** as part of the driver name. The default ODBC driver, IBM DB2 ODBC DRIVER, is set to the Default DB2 copy. The name of the driver for each installation is "IBM DB2 ODBC DRIVER - <DB2 Copy Name>".

Note:

- Only one DB2 copy can be used by the same ODBC application.
- Even when you set up a Data source with the default ODBC driver, it will be configured to access the DB2 copy that was the default at the time the Data source was cataloged.
- If you move or migrate instances from one DB2 copy to another, you will need to reconfigure the associated Data sources.

DB2 .NET Data Provider

The DB2 .NET Data Provider is not accessed by the **DB2 copy Name**. Instead, depending on the version of the provider that the application requires, it finds that version and uses it using the standard methods.

JDBC/SQLJ

JDBC uses the current version of the driver in the classpath. The Type 2 JDBC driver uses the native DLL. By default, the classpath is configured to point to the default DB2 copy. Running db2envar.bat from the DB2 copy you want to use will update your PATH and CLASSPATH settings for this copy.

MMC Snap-in

The MMC Snap-in launches the DB2 Control Center for the Default DB2 copy.

WMI WMI does not support multiple DB2 copies. You can register only one copy of WMI at a time. To register WMI, follow this process:

- Unregister the WMI Schema extensions.
- Unregister the COM object.
- Register the new COM object.
- Use MOFCOMP to extend the WMI schema.

WMI is not registered during DB2 installation. You still need to complete the two registration steps. WMI is a selectable feature in DB2 products, in PE and above. It is not selected by default, nor is it in the typical install.

CLI applications

CLI applications that dynamically load the DB2 client libraries should use the **LoadLibraryEx** API with the LOAD_WITH_ALTERED_SEARCH_PATH option, instead of the LoadLibrary option. If you do not use the LoadLibrary option, you will need to specify db2app.dll in the Path by running **db2envar.bat** from the bin directory of the DB2 copy that you

want to use. For applications that link using db2apie.lib, to use a different DB2 copy, you can use the /delayload option in your **link** command to delay load db2app.dll and call the **db2SelectDB2Copy** API prior to any DB2 calls.

DB2 System Tray

To reduce the number of system tray executables running on the system, by default any system tray's that are running in the previous Default DB2 copy when the default copy is changed are disabled.

Related concepts:

- "Multiple DB2 copies on the same computer (Windows)" on page 17

Related tasks:

- "Changing the Default DB2 copy after installation (Windows)" on page 21
- "Setting the DAS when running multiple DB2 copies (Windows)" on page 24

Related reference:

- "Multiple DB2 copies roadmap" on page 15

Setting the DAS when running multiple DB2 copies (Windows)

In DB2 Version 9, you can have multiple DB2 copies running on the same computer. This affects how the DB2 Administration Server (DAS) operates. The DAS is a unique component within DB2 that is limited to having only one version active, despite how many DB2 copies are installed on the same computer. For this reason the following restrictions and functional requirements apply.

On the server, there can be only one DAS version and it administers instances as follows:

- If the DAS is on Version 9, then it can administrator Version 8 and Version 9 instances.
- If the DAS is on Version 8, then it can administer only Version 8 instances. You can migrate your Version 8 DAS, or drop it and create a new Version 9 DAS to administer the Version 8 and Version 9 instances. This is required only if you want to use the Control Center to administer the instances.

Restrictions:

Only one DAS can be created on a given computer at any given time despite the number of DB2 copies that are installed on the same computer. This DAS will be used by all the DB2 copies that are on the same computer. In Version 9 or later, the DAS can belong to any DB2 copy that is currently installed.

Procedure:

To move the DAS from one DB2 Version 9 copy to another DB2 Version 9 copy, use the `dasupdt - Update DAS` command.

You can also use this command when you need to move the DB2 Administration Server (DAS) to a new Default DB2 copy in the same version.

Note:

- The **dasupdt** command can only be used to move the DAS between various DB2 copies of the same DB2 release (that is, between different Fix Packs). It cannot be used to setup DAS.
- For migration from Version 8 to Version 9 DAS, use the **dasmigr** command.
- If DAS is not set up, then a regular DAS setup procedure should be followed to set it up on one of the DB2 copies.

Related concepts:

- “DB2 administration server (DAS) configuration on Enterprise Server Edition (ESE) systems” on page 106
- “Multiple DB2 copies on the same computer (Windows)” on page 17
- “Security considerations for the DB2 administration server (DAS) on Windows” on page 102

Related tasks:

- “Changing the Default DB2 copy after installation (Windows)” on page 21
- “Configuring the DB2 administration server (DAS)” on page 95
- “Creating a DB2 administration server (DAS)” on page 93
- “Listing the DB2 administration server (DAS)” on page 95
- “Removing the DB2 administration server (DAS)” on page 103
- “Setting up DB2 administration server (DAS) with Enterprise Server Edition (ESE) systems” on page 104
- “Starting and stopping the DB2 administration server (DAS)” on page 94
- “Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration” on page 96

Related reference:

- “dasmigr - Migrate the DB2 administration server command” in *Command Reference*
- “dasupdt - Update DAS command” in *Command Reference*
- “Multiple DB2 copies roadmap” on page 15

Setting the default instance when using multiple DB2 copies (Windows)

In Version 9.1, the DB2INSTANCE environment is set according to the DB2 copy that your environment is currently set up to use. If you do not set it explicitly to another instance in the current copy, it defaults to the default instance that is specified with the DB2INSTDEF profile registry variable.

Note: DB2INSTDEF is the default instance variable that is specific to the current DB2 copy in use (that is, every DB2 copy has its own DB2INSTDEF). DB2INSTANCE is set to the current instance you are using.

- If DB2INSTANCE is not set for a particular DB2 copy, then the value of DB2INSTDEF is used for that DB2 copy.
- DB2INSTANCE is only valid for instances under the DB2 copy that you are using. However, if you switch copies by running the **db2envar.bat** command, DB2INSTANCE will be updated to the value of DB2INSTDEF for the DB2 copy that you switched to initially.

All global profile registry variables are specific to a DB2 copy, unless you specify them using **SET VARIABLE=<variable_name>**.

Procedure:

To set the default instance, you can set the DB2INSTDEF profile registry variable using the **db2set** command. When you access a different DB2 copy, you do not have to change the value of DB2INSTANCE.

Related concepts:

- “Environment variables and the profile registry” on page 65
- “Multiple DB2 copies on the same computer (Windows)” on page 17

Related tasks:

- “Client connectivity using multiple DB2 copies (Windows)” on page 22
- “Setting environment variables on Windows” on page 77
- “Setting the current instance environment variables” on page 67
- “Setting the DAS when running multiple DB2 copies (Windows)” on page 24

Related reference:

- “General registry variables” on page 70
- “Multiple DB2 copies roadmap” on page 15
- “db2set - DB2 profile registry command” in *Command Reference*

Managing DB2 copies (Windows)

When updating your DB2 product, you will be required to specify whether you want to update an existing DB2 copy, or whether to install a new one. You must select the option work with existing to update a DB2 copy. You will not be able to update more than one DB2 copy at the same time. In order to update other DB2 copies that may be installed on the same computer, you need to rerun the installation.

The installation provides the option to migrate DB2 Version 8 (in the same path) or to install a new DB2 Version 9 Copy without modifying the DB2 Version 8 installation. If you select to migrate, your Version 8 installation will be removed. If you select to install a new DB2 copy, you can later choose to migrate your instances using the **db2ckmig** and **db2imigr** commands.

You can use the **db2iupdt** command to move a DB2 instance between different Version 9 DB2 copies, and the **db2imigr** command to move a Version 8 instance to Version 9. See Migrating a DB2 server (Windows) for complete details on how to migrate to DB2 Version 9.

Note:

- Coexistence of DB2 Version 7 and DB2 Version 9 is not supported.
- Coexistence of a 32-bit DB2 and a 64-bit DB2 on the same Windows X64 computer is not supported.

It is not possible to migrate from a 32-bit X64 DB2 installation at Version 8 to a 64-bit installation at Version 9. Instead, you need to migrate to Version 9 32-bit to use the X64 DB2 installation to move to 64-bit. The 32-bit version will be removed. If you have more than one 32-bit DB2 copy installed, you will need to move all of your instances to one DB2

copy and remove these copies from the computer. For more information, see Migrating DB2 32-bit servers to 64-bit systems (Windows).

- To move an instance from one DB2 Version 9 copy to another, you can use the **db2iupdt** command.
- If you use the **db2imigr** command to migrate your instances from Version 8, you will need to reconfigure any ODBC data sources.

In summary, on Windows:

- If you have multiple DB2 Version 9 copies, the installation options are install a new copy or work with an existing DB2 copy, which you can upgrade or add new features. The migrate option will only show if you also have a DB2 UDB Version 8 copy in addition to the DB2 Version 9 copies.
- If DB2 UDB Version 8 is installed, the installation options are migrate the existing Version 8 copy or install a new DB2 copy.
- If DB2 Version 7 or earlier is installed, the installation displays a message to indicate that migration to DB2 Version 9 is not supported. You can only install a new DB2 copy after uninstalling Version 7. In other words, Version 7 and Version 9 cannot coexist.

Related concepts:

- “Multiple DB2 copies on the same computer (Windows)” on page 17

Related tasks:

- “Migrating a DB2 server (Linux and UNIX)” in *Migration Guide*
- “Migrating a DB2 server (Windows)” in *Migration Guide*
- “Migrating DB2 32-bit servers to 64-bit systems (Windows)” in *Migration Guide*
- “Running multiple instances concurrently (Windows)” on page 27

Related reference:

- “db2ckmig - Database pre-migration tool command” in *Command Reference*
- “db2imigr - Migrate instance command” in *Command Reference*
- “db2iupdt - Update instances command” in *Command Reference*
- “Multiple DB2 copies roadmap” on page 15

Running multiple instances concurrently (Windows)

You can run multiple instances concurrently in the same DB2 copy, or in different DB2 copies.

Procedure:

To run multiple instances concurrently in the same DB2 copy, use either of the following methods:

- Using the Control Center:

- | |
|--|
| <ol style="list-style-type: none">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:) 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.

To run multiple instances concurrently in different DB2 copies, use either of the following methods:

- Using the DB2 command window from the Start → Programs → IBM DB2 → <DB2 Copy Name> → Command Line Tools → DB2 Command Window: the command window is already set up with the correct environment variables for the particular DB2 copy chosen.
- Using db2envvar.bat from a command window:
 1. Open a command window.
 2. Run the db2envvar.bat file using the fully qualified path for the DB2 copy that you want the application to use:

```
<DB2 Copy install dir>\bin\db2envvar.bat
```

After you switch to a particular DB2 copy, use the method specified in the section above, "To run multiple instances concurrently in the same DB2 copy", to start the instances.

Related concepts:

- "Multiple instances of the database manager" on page 16

Related tasks:

- "Creating additional instances" on page 38
- "Managing DB2 copies (Windows)" on page 26
- "UNIX details when creating instances" on page 39
- "Windows details when creating instances" on page 40

Related reference:

- "Multiple DB2 copies roadmap" on page 15
- "db2envvar.bat command" in *Command Reference*
- "db2start - Start DB2 command" in *Command Reference*

Removing DB2 copies (Linux, UNIX, and Windows)

Procedure:

To uninstall DB2 copies on Linux and UNIX, use the **db2_deinstall** command from the DB2 copy that you are using. This command uninstalls installed DB2 products or features that are in the same install path as the **db2_deinstall** tool. Use the **db2ls** command to see the list of installed DB2 products and features. If one or more instances are currently associated with a DB2 copy, that DB2 copy cannot be uninstalled.

To uninstall DB2 copies on Windows operating systems, use one of the following methods:

- You can uninstall any DB2 copy by using the Windows Add/Remove Control Panel Applet. The Default DB2 copy will have the word (**default**) appended to it.
- Run the **db2unins** command from the installed DB2 copy directory

Note: On Windows:

- You can uninstall DB2 even when there are instances associated with DB2 copies. If you do this, the instance information will be removed with the DB2 uninstall. Therefore, take extra care when managing, recovering, and uninstalling instances.
- If multiple Version 9 copies are installed, you cannot remove the default DB2 copy. If you want to remove the default DB2 copy, you will need to switch the default DB2 copy to one of the other DB2 copies prior to uninstalling. For more information on switching the default DB2 copy, see the **db2swtch** command.

Related concepts:

- “Multiple DB2 copies on the same computer (Windows)” on page 17
- “Multiple DB2 copies on the same computer (Linux and UNIX)” in *Installation and Configuration Supplement*

Related reference:

- “Multiple DB2 copies roadmap” on page 15
- “db2_deinstall - Uninstall DB2 products or features command” in *Command Reference*
- “db2ls - List installed DB2 products and features command” in *Command Reference*
- “db2swtch - Switch default DB2 copy command” in *Command Reference*
- “db2unins - Uninstall DB2 database product command” in *Command Reference*

Working with partitioned databases

This section describes various aspects of partitioned databases that you need to be aware of before creating a database.

Management of database server capacity

If database manager capacity does not meet your present or future needs, you can expand its capacity in the following ways:

- Add disk space and create additional containers.
- Add memory.

If these simple strategies do not add the capacity you need, consider the following methods:

- Add processors.

If a single-partition database configuration with a single processor is used to its maximum capacity, you might either add processors or add database partitions. The advantage of adding processors is greater processing power. In an SMP system, processors share memory and storage system resources. All of the processors are in one system, so there are no additional overhead considerations such as communication between systems and coordination of tasks between systems. Utilities in DB2 such as load, backup, and restore can take advantage of the additional processors. DB2 database supports this environment.

Note: Some operating systems, such as the Solaris operating system, can dynamically turn processors on- and off-line.

If you add processors, review and modify some database configuration parameters that determine the number of processors used. The following database configuration parameters determine the number of processors used and might need to be updated:

- Default degree (dft_degree)
- Maximum degree of parallelism (max_querydegree)
- Enable intra-partition parallelism (intra_parallel)

You should also evaluate parameters that determine how applications perform parallel processing.

In an environment where TCP/IP is used for communication, review the value for the DB2TCPCONNMGERS registry variable.

- Add physical partitions.

If your database manager is currently in a partitioned database environment, you can increase both data-storage space and processing power by adding separate single-processor or multiple-processor physical partitions. The memory and storage system resources on each database partition are not shared with the other database partitions. Although adding database partitions might result in communication and task-coordination issues, this choice provides the advantage of balancing data and user access across more than one system. DB2 database supports this environment.

You can add database partitions either while the database manager system is running or while it is stopped. If you add database partitions while the system is running, however, you must stop and restart the system before databases migrate to the new database partition.

When you scale your system by changing the environment, you should be aware of the impact that such a change can have on your database procedures such as loading data, backing up the database, and restoring the database.

When you add a new database partition, you cannot drop or create a database that takes advantage of the new database partition until the procedure is complete, and the new server is successfully integrated into the system.

Related concepts:

- “Adding database partitions in a partitioned database environment” on page 123

Multiple logical partitions

When several database partition servers are running on the same computer, the computer is said to be running multiple logical partitions. This section describes when to use and how to configure multiple logical partitions.

When to use multiple logical partitions

Typically, you configure DB2 Enterprise Server Edition to have one database partition server assigned to each computer. There are several situations, however, in which it would be advantageous to have several database partition servers running on the same computer. This means that the configuration can contain more database partitions than computers. In these cases, the computer is said to be running *multiple logical partitions* or *multiple logical nodes* if they participate in the *same* instance. If they participate in different instances, this computer is *not* hosting multiple logical partitions.

With multiple logical partition support, you can choose from three types of configurations:

- A standard configuration, where each computer has only one database partition server.
- A multiple logical partition configuration, where a computer has more than one database partition server.
- A configuration where several logical partitions run on each of several computers.

Configurations that use multiple logical partitions are useful when the system runs queries on a computer that has symmetric multiprocessor (SMP) architecture. The ability to configure multiple logical partitions on a computer is also useful if a computer fails. If a computer fails (causing the database partition server or servers on it to fail), you can restart the database partition server (or servers) on another computer using the DB2START NODENUM command. This ensures that user data remains available.

Another benefit is that multiple logical partitions 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 partitions” on page 31

Related reference:

- “db2start - Start DB2 command” in *Command Reference*

Configuring multiple logical partitions

Procedure:

You can configure multiple logical partitions in one of two ways:

- Configure the logical partitions (database partitions) in the `db2nodes.cfg` file. You can then start all the logical and remote partitions with the DB2START command or its associated API.

Note: For Windows, you must use `db2ncrt` to add a database partition if there is no database in the system; or, DB2START ADDNODE command if there is one or more databases. Within Windows, the `db2nodes.cfg` file should never be manually edited.

- Restart a logical partition on another processor on which other logical partitions (nodes) are already running. This allows you to override the hostname and port number specified for the logical partition in `db2nodes.cfg`.

To configure a logical partition (node) in `db2nodes.cfg`, you must make an entry in the file to allocate a logical port number for the database partition. Following is the syntax you should use:

```
nodenumber hostname logical-port netname
```

Note: For Windows, you must use `db2ncrt` to add a database partition if there is no database in the system; or, DB2START ADDNODE command if there is one or more databases. Within Windows, the `db2nodes.cfg` file should never be manually edited.

The format for the *db2nodes.cfg* file on Windows is different when compared to the same file on Unix. On Windows, 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 might 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 partitions” on page 30

Related tasks:

- “Changing node and database configuration files” on page 279
- “Creating a node configuration file” on page 81

Related reference:

- “db2ncrt - Add database partition server to an instance command” in *Command Reference*
- “db2start - Start DB2 command” in *Command Reference*

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 database partition’s *services* file of the *etc* directory as shown below. The FCM uses the specified port to communicate. If you have defined multiple database partitions on the same host, you must define a range of ports as shown below.

Before attempting to manually configure memory for the fast communications manager (FCM), it is recommended that you start with the automatic setting, which is also the default setting, for the number of FCM Buffers (*fcnum_buffers*) and for the number of FCM Channels (*fcnum_channels*). Use the system monitor data for FCM activity to determine if this setting is appropriate.

Windows Considerations

If you are using DB2 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 database partition
- The **db2icrt** utility when it creates a new instance
- The **db2ncrt** utility when it adds the first database partition on the computer

The syntax of a service entry is as follows:

```
DB2_instance port/tcp #comment
```

DB2_instance

The value for *instance* is the name of the database manager instance. All

characters in the name must be lowercase. Assuming an instance name of db2puser, you would specify DB2_db2puser

port/tcp

The TCP/IP port that you want to reserve for the database partition.

#comment

Any comment that you want to associate with the entry. The comment must be preceded by a pound sign (#).

If the 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 database instance are the same in all services files of the etc directory (though other entries that do not apply to your partitioned database environment 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.

Due to the way the FCM infrastructure utilizes TCP sockets and directs network traffic, FCM users on AIX 5.x should set the kernel parameter "tcp_nodelayack" to 1.

Related concepts:

- "Database partition and processor environments" in *Administration Guide: Planning*
- "Aggregate registry variables" on page 75
- "The FCM buffer pool and memory requirements" in *Performance Guide*

Related reference:

- "MPP configuration variables" in *Performance Guide*

Preparing to create a database

Based on your business needs and environment, there are many concepts and tasks that 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.

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 on the same physical server providing a unique database server environment for each 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.

Multiple instances will require:

- Additional system resources (virtual memory and disk space) for each instance.
- More administration 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 database processes.

Terminology:

Bit-width

The number of bits used to address virtual memory: 32-bit and 64-bit are the most common. This term might be used to refer to the bit-width of an instance, application code, external routine code. 32-bit application means the same things as 32-bit width application.

32-bit DB2 instance

A DB2 instance that contains all 32-bit binaries including 32-bit shared libraries and executables.

64-bit DB2 instance

A DB2 instance that contains 64-bit shared libraries and executables, and also all 32-bit client application libraries (included for both client and server), and 32-bit external routine support (included only on a server instance).

You can:

- Create an instance.
- Drop an instance.
- Start an instance.
- Stop an instance.
- Attach to an instance.

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 the DB2 database product was installed.

In a partitioned database environment, 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 computers in the instance can access.

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

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

- The environment variable `DB2INSTANCE` is set to “DB2”.
- The DB2 registry variable `DB2INSTDEF` is set to “DB2”.

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

On Windows, the instance name is the same as the name of the service, so it should not conflict. No instance name should be the same as another service name. 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, the database environment for each user must be updated so that it can access an instance and run the DB2 database 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. Use `sqllib/userprofile` and `sqllib/usercshrc` to customize a script for each user.

The blank files `sqllib/userprofile` and `sqllib/usercshrc` are created during instance creation to allow you to add your own instance environment settings. The `db2profile` and `db2cshrc` files are overwritten during an instance update in a DB2 FixPak installation. If you do not want the new environment settings in the

db2profile or db2cshrc scripts, you can override them using the corresponding *user* script, which is called at the end of the db2profile or db2cshrc script. During an instance migration (using the **db2imigr** command), the *user* scripts are copied over so that your environment modifications will still be in use.

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 sql1ib subdirectory of the instance owner's home directory.
- Set the DB2INSTANCE environment variable to the instance name.

Related concepts:

- "Multiple instances on a Linux or UNIX operating system" on page 36
- "Multiple instances on a Windows operating system" on page 37
- "About authorities" in *Administration Guide: Planning*
- "About configuration parameters" in *Administration Guide: Planning*
- "About databases" in *Administration Guide: Planning*
- "About the database manager" in *Administration Guide: Planning*

Related tasks:

- "Adding instances" on page 41
- "Auto-starting instances" on page 42
- "Creating additional instances" on page 38
- "Listing instances" on page 41
- "Running multiple instances concurrently (Windows)" on page 27
- "Setting the current instance environment variables" on page 67
- "UNIX details when creating instances" on page 39
- "Windows details when creating instances" on page 40

Instance management

This section contains additional concepts and tasks related to instance management.

Multiple instances on a Linux or UNIX operating system

It is possible to have more than one instance on a UNIX operating system. However, you can only work within one instance of the DB2 database manager 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 the DB2 database manager.

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 might 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 39

Multiple instances on a Windows operating system

It is possible to run multiple instances of the DB2 database manager on the same computer. Each instance of the DB2 database manager maintains its own databases and has its own database manager configuration parameters.

An instance of the DB2 database manager 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 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 the DB2 database manager 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_computer\SOFTWARE\IBM\DB2\PROFILES\

You can run multiple DB2 database instances concurrently, in the same DB2 copy or in different DB2 copies.

- To work with an instance in the same DB2 copy, 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.

- To work with an instance in different DB2 copies, use either of the following methods:
 - Using the DB2 command window from the Start → Programs → IBM DB2 → <DB2 Copy Name> → Command Line Tools → DB2 Command Window: the command window is already set up with the correct environment variables for the particular DB2 copy chosen.
 - Using db2envvar.bat from a command window:
 1. Open a command window.
 2. Run the db2envvar.bat file using the fully qualified path for the DB2 copy that you want the application to use:


```
<DB2 Copy install dir>\bin\db2envvar.bat
```

Related concepts:

- “High availability” in *Data Recovery and High Availability Guide and Reference*

Related tasks:

- “Windows details when creating instances” on page 40

Creating additional instances

Although an instance is created as part of the installation of the DB2 database manager, your business needs might 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 database instances. The computer where you add the instance becomes the instance-owning computer (node zero). Ensure that you add instances on a computer 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 DB2 instance, 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 Enterprise Server Edition, 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 database partition, and working with Fast Communication Manager (FCM), you can have multiple connections between database 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 considerations for remote clients” on page 495
- “Authentication methods for your server” on page 490

Related reference:

- “db2icrt - Create instance command” in *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 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 a DB2 client. For other DB2 products, this is a required parameter.

Note: FencedID might 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 server, you can use the following command:
`db2icrt -u db2fenc1 db2inst1`
- If you installed the DB2 Connect™ Enterprise Server Edition only, you can use the instance name as the Fenced ID also:
`db2icrt -u db2inst1 db2inst1`
- To add an instance for a DB2 client, you can use the following command:
`db2icrt db2inst1 -s client -u fencedID`

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

Related reference:

- “`db2icrt - Create instance command`” in *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 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 computers in the partition database system.

The following example could be used, on DB2 Enterprise Server Edition for Windows:

```
db2icrt inst1 -s ese
-p:\\computerA\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 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 *Command Reference*

Adding instances

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.

Procedure:

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. From 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 34

Related tasks:

- “Listing instances” on page 41

Listing instances

Use the Control Center or the **db2ilist** command to get a list of instances, as follows:

- On Version 8 or earlier, all the instances on the system are listed.
- On Version 9 or later, only the instances from the DB2 copy where the **db2ilist** command is invoked from are listed.

Procedure:

To get a list of instances 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.

To get a list of instances 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 *Command Reference*

Auto-starting instances

Procedure:

On Windows operating systems, the DB2 database 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 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 34

Related reference:

- “db2iauto - Auto-start instance command” in *Command Reference*

Quiescing and unquiescing instances

You can use the Quiesce window to force users off an instance, except the user or group that you specify. You can use the **Unquiesce** menu option to return an instance to an active state so that all users can access the instance.

Prerequisites:

To quiesce or unquiesce an instance, you must have either SYSADM, SYSCTRL, or SYSMAINT authority.

Procedure:

To quiesce an instance using the Control Center:

1. Open the Quiesce window: Expand the object tree until you find the instance that you want to quiesce. Right-click the instance and select **Quiesce** from the pop-up menu. The Quiesce window opens.
2. Specify whether you want to allow a user or a group to access the instance. If you are allowing a user to attach to the instance, use the **User** controls to specify a specific user. If you are allowing a group to attach to the instance, use the **Group** controls to specify a specific group.

When you click **OK**, the Quiesce window closes and the instance is quiesced. Only the specified user or group will be able to attach to the quiesced instance until either the instance is unquiesced or stopped.

To unquiesce an instance using the Control Center:

1. Expand the object tree until you find the instance that you want to unquiesce.
2. Right-click the instance and select **Unquiesce** from the pop-up menu. The instance will be unquiesced immediately.

Related reference:

- “QUIESCE command” in *Command Reference*
- “UNQUIESCE command” in *Command Reference*

Setting the DB2 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 want 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 environment manually on UNIX” on page 44

Setting the DB2 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 want 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 environment automatically on UNIX” on page 43

Automatic client rerouting

This section describes the configuration and maintenance of automatic client rerouting, and also includes troubleshooting information.

Automatic client reroute roadmap

Automatic client reroute is a DB2 Database for Linux, UNIX, and Windows feature that allows client applications to recover from a loss of communication with the server so that the application can continue its work with minimal interruption. Automatic client reroute can be accomplished only if an alternate server has been specified prior to the loss of communication.

Table 2 lists the relevant topics in each category.

Table 2. Roadmap to automatic client reroute information

Category	Related topics
General information	<ul style="list-style-type: none">• Automatic client reroute• Automatic client reroute limitations• Automatic client reroute description and setup
Configuration	<ul style="list-style-type: none">• Specifying a server for automatic client reroute• Automatic client reroute configuration (DB2_MAX_CLIENT_CONNRETRIES and DB2_CONNRETRIES_INTERVAL)• Client reroute setup when using JCC Type 4 drivers
Examples	<ul style="list-style-type: none">• Automatic client reroute examples
Interaction with other DB2 features	<ul style="list-style-type: none">• Automatic client reroute and high availability disaster recovery (HADR)• Interaction between client connection timeout and client reroute• IBM DB2 Driver for JDBC and SQLJ client reroute support
Troubleshooting	<ul style="list-style-type: none">• Distributor considerations

Automatic client reroute description and setup

The main goal of the automatic client reroute feature is to enable a DB2 database 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 connection.

The automatic client reroute feature could be used within the following configurable environments:

1. Enterprise Server Edition (ESE) with the database partitioning feature (DPF)
2. DataPropagator™ (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 the case of the DB2 Connect server, because there is no requirement for the synchronization of local databases, you only need to ensure that both the original and alternate DB2 Connect servers have the target host or iSeries™ database catalogued in such a way that it is accessible using an identical database alias.

In order for the DB2 database system to have the ability to recover from a loss of communications, an alternative server location must be specified before the loss of communication occurs. 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. 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 database partition 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 database partition N1 (on the server instance):

```
db2 update alternate server for database db2 using hostname AAA port 123
```

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 client coded will attempt to re-establish the connection by using the alternate server information. The DB2 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.

Consider the following two items involving alternate server connectivity with DB2 Connect server:

- The first consideration involves using DB2 Connect server for providing access to a host or iSeries database on behalf of both remote and local clients. In such situations, confusion can arise regarding alternate server connectivity information in a system database directory entry. To minimize this confusion, consider cataloging two entries in the system database directory to represent the same host or iSeries database. Catalog one entry for remote clients and catalog another for local clients.
- Secondly, the alternate server information that is returned from a target server is kept only in cache. If the DB2 process is terminated, the cache information, therefore the alternate server information, is lost.

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 47

- “Client reroute setup when using JCC Type 4 drivers” on page 54

Related reference:

- “Automatic client reroute examples” on page 49
- “Automatic client reroute roadmap” on page 45

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 database 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 database is set up for a loopback, TCP/IP communications protocol must be used in order accommodate the automatic client reroute feature.
- When cataloging on a DB2 Connect server and you have an environment where you want automatic client rerouting to be done, you will have situations which have implications:
 - When using DB2 Connect server for providing access to a host or iSeries database on behalf of both remote and local clients. Confusion can arise regarding alternate server connectivity information in a system database directory entry. To minimize this confusion, consider cataloging two entries in the system database directory to represent the same host or iSeries database. Catalog one entry for remote clients and catalog another for local clients.
 - When the alternate server information that is returned from a target server is kept only in cache memory. If the DB2 database process is terminated, the cache information, and therefore the alternate server information, is lost.
- 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 Database for Linux, UNIX, and Windows supports the automatic client reroute feature for both the client and the server if both the client and server support this feature. Other DB2 database product families do not currently support this feature.
- The behavior of the automatic client reroute feature and the behavior of rerouting in a DB2 Universal Database (DB2 UDB) for z/OS sysplex environment are somewhat different. Specifically:
 - The automatic client reroute feature requires the primary server to designate a single alternative server. This is done using the **UPDATE ALTERNATE SERVER FOR DATABASE** or **UPDATE ALTERNATE SERVER FOR LDAP DATABASE** command issued at the primary server. This command updates the local database directory with the alternate server information so that other applications at the same client have access this information. By contrast, a data-sharing sysplex used for DB2 UDB for z/OS maintains, in memory, a list

of one or more servers to which the client can connect. If a communication failure happens, the client uses that list of servers to determine the location of the appropriate alternative server.

- In the case of the automatic client reroute feature, the server informs the client of the most current special register settings whenever a special register setting is changed. This allows the client, to the best of its ability, to re-establish the runtime environment after a reroute has occurred. By contrast, a sysplex used for DB2 UDB for z/OS does not provide the special register settings to the client and therefore, the client must reinstate the runtime environment after the reroute is complete.

As of FixPak 7, full automatic client reroute support is available only between a Linux, UNIX, or Windows client and a Linux, UNIX, or Windows server. It is not available between a Linux, UNIX, or Windows client and a DB2 UDB for z/OS sysplex server (any supported version); only the reroute capability is supported.

- The DB2 database server installed in the alternate host server must be the same version (but could have a higher FixPak) when compared to the DB2 database instance 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 the DB2 database 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 and XQuery 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.

- Do not run high availability disaster recovery (HADR) commands on client reroute-enabled database aliases. HADR commands are implemented to identify the target database using database aliases. Consequently, if the target database has an alternative database defined, it is difficult for HADR commands to determine the database on which the command is actually operating. While a client might need to connect using a client reroute-enabled alias, HADR

commands must be applied on a specific database. To accommodate this, you can define aliases specific to the primary and standby databases and only run HADR commands on those aliases.

An alternate way to implement automatic client rerouting is to use the DNS entry to specify an alternate 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 database system would alternate between the IP addresses for the DNS entry.

Related tasks:

- “Specifying a server for automatic client reroute” on page 49

Related reference:

- “Automatic client reroute roadmap” on page 45
- “UPDATE ALTERNATE SERVER FOR DATABASE command” in *Command Reference*
- “UPDATE ALTERNATE SERVER FOR LDAP DATABASE command” in *Command Reference*

Specifying a server for automatic client reroute

Whenever a DB2 server or DB2 Connect 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 client code attempts to re-establish the connection to the original server which might 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** or **UPDATE ALTERNATE SERVER FOR LDAP DATABASE** command. These commands update the alternate server information for a database alias in the system database directory.

Related concepts:

- “Automatic client reroute description and setup” on page 45

Related reference:

- “UPDATE ALTERNATE SERVER FOR DATABASE command” in *Command Reference*
- “UPDATE ALTERNATE SERVER FOR LDAP DATABASE command” in *Command Reference*
- “Automatic client reroute roadmap” on page 45

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, the DB2 database system will try to establish the connection to host “hornet” (with port 456) again. If it is still not working, the DB2 database system 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 standby database is also created at host “montero” with port 456. Information on how to setup HADR for both a primary and standby 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, the DB2 database system will try to establish the connection to host “hornet” (with port 456) again. If it is still not working, the DB2 database system 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 45

Related tasks:

- “Specifying a server for automatic client reroute” on page 49

Related reference:

- “Automatic client reroute roadmap” on page 45

Automatic client reroute configuration (DB2_MAX_CLIENT_CONNRETRIES and DB2_CONNRETRIES_INTERVAL)

By default, the automatic client reroute feature retries the connection to a database repeatedly for up to 10 minutes. It is, however, possible to configure the exact retry behavior using one or both of the following two registry variables:

- **DB2_MAX_CLIENT_CONNRETRIES**: The maximum number of connection retries attempted by automatic client reroute.
- **DB2_CONNRETRIES_INTERVAL**: The sleep time between consecutive connection retries, in number of seconds.

If **DB2_MAX_CLIENT_CONNRETRIES** is set, but **DB2_CONNRETRIES_INTERVAL** is not, **DB2_CONNRETRIES_INTERVAL** defaults to 30.

If **DB2_MAX_CLIENT_CONNRETRIES** is not set, but **DB2_CONNRETRIES_INTERVAL** is set, **DB2_MAX_CLIENT_CONNRETRIES** defaults to 10.

If neither **DB2_MAX_CLIENT_CONNRETRIES** nor **DB2_CONNRETRIES_INTERVAL** is set, the automatic client reroute feature reverts to its default behavior described previously.

Note:

Users of Type 4 connectivity with the DB2 Universal JDBC Driver should use the following two datasource properties to configure automatic client rerouting:

- **maxRetriesForClientReroute**: Use this property to limit the number of retries if the primary connection to the server fails. This property is only used if the **retryIntervalClientReroute** property is also set.
- **retryIntervalForClientReroute**: Use this property to specify the amount of time (in seconds) to sleep before retrying again. This property is only used if the **maxRetriesForClientReroute** property is also set.

Related reference:

- “Automatic client reroute roadmap” on page 45

Interaction between client connection timeout and client reroute

For CLI/ODBC, OLE DB, and ADO.NET applications, you can set a connection timeout value to specify the number of seconds that the client application waits for a reply when trying to establish a connection to a server before terminating the connection attempt and generating a communication timeout.

If client reroute is enabled, you need to set the connection timeout value to a value that is equal to or greater than the maximum time it takes to connect to the server. Otherwise, the connection might timeout and be rerouted to the alternate server by client reroute. For example, if on a normal day it takes about 10 seconds to connect to the server, and on a busy day it takes about 20 seconds, the connection timeout value should be set to at least 20 seconds.

Related concepts:

- “Client reroute” in *Administration Guide: Planning*

Related reference:

- “ConnectTimeout CLI/ODBC configuration keyword” in *Call Level Interface Guide and Reference, Volume 1*
- “Automatic client reroute roadmap” on page 45

Distributor considerations

When a client to server connection fails, the client’s requests for reconnection are distributed to a defined set of systems by a distributor or dispatcher, such as WebSphere® EdgeServer.

You might be using distributor technology in an environment similar to the following:

Client → distributor technology → (DB2 Connect Server 1 or DB2 Connect Server 2) → DB2 z/OS

where:

- The distributor technology component has a TCP/IP host name of **DHostname**
- The DB2 Connect Server 1 has a TCP/IP host name of **GWYhostname1**
- The DB2 Connect Server 2 has a TCP/IP host name of **GWYhostname2**
- The DB2 z/OS server has a TCP/IP host name of **zOShostname**

The client is catalogued using **DHostname** in order to utilize the distributor technology to access either of the DB2 Connect Servers. The intervening distributor technology makes the decision to use **GWYhostname1** or **GWYhostname2**. Once the decision is made, the client has a direct socket connection to one of these two DB2 Connect gateways. Once the socket connectivity is established to the chosen DB2 Connect server, you have a typical client to DB2 Connect server to DB2 z/OS connectivity.

For example, assume the distributor chooses **GWYhostname2**. This produces the following environment:

Client → DB2 Connect Server 2 → DB2 z/OS

The distributor does not retry any of the connections if there is any communication failure. If you want to enable the automatic client reroute feature for a database in such an environment, the alternative server for the associated database or databases in the DB2 Connect server (DB2 Connect Server 1 or DB2 Connect Server 2) should be set up to be the distributor (**DHostname**). Then, if DB2 Connect Server 1 locks up for any reason, automatic client rerouting is triggered and a client connection is retried with the distributor as both the primary and the alternate server. This option allows you to combine and maintain the distributor capabilities with the DB2 automatic client reroute feature. Setting the alternate server to a host other than the distributor host name still provides the clients with the automatic client reroute feature. However, the clients will establish direct connections to the defined alternate server and bypass the distributor technology, which eliminates the distributor and the value that it brings.

The automatic client reroute feature intercepts the following SQL codes:

- sqlcode -20157
- sqlcode -1768 (reason code = 7)

Note: Client reroute might not be informed of socket failures in a timely fashion if the setting of the "TCP Keepalive" operating system configurations parameter is too high. (Note that the name of this configuration parameter varies by platform.)

Related reference:

- "Automatic client reroute roadmap" on page 45

Client reroute setup when using JCC Type 4 drivers

Implementing client reroute while using JCC Type 4 drivers requires a different setup. JCC Type 4 clients do not require a DB2 installation on the client side server. Instead of allowing a client application to pick up connection information from the local database directory, the server name and port are explicitly included in the JCC connection attempt. If, however, at the time of connection, the server function has been taken over by an alternate server, the client will not only be unable to connect, but will not know where to find the alternate server information.

The best way to avoid this is to set up an application to retrieve the alternate server information. By using the `javax.sql.DataSource` interface, alternate server parameters can be picked up by the JCC application and kept in non-volatile storage on the client machine. The storage can be done using the JNDI API. If, for instance, a local file system is specified as the non-volatile storage, JNDI will create a `.bindings` file which will contain the required alternate server information. After the current JVM is shut down, the information will then persist in that file until a new JVM is created. The new JVM will attempt to connect to the server. If the alternate server information has been updated, this will be updated on the client machine without requiring your intervention. If the server is missing however, the `.binding` file will be read and a new connection attempt will be made at the location of the alternate server. LDAP can also be used to provide non-volatile storage for the alternate server information. Using volatile storage is not recommended, as a client machine failure could result in the loss of alternate server data stored in memory.

Related concepts:

- "Automatic client reroute description and setup" on page 45
- "Automatic client reroute limitations" on page 47

Related reference:

- "Automatic client reroute roadmap" on page 45

Automatic storage

This section contains information about automatic storage databases and table spaces.

Automatic storage databases

An automatic storage database is one in which table spaces can be created and whose container and space management characteristics are completely determined by the DB2 database manager. At the most basic level, databases that are enabled for automatic storage have a set of one or more storage paths associated with them. A table space can be defined as "managed by automatic storage" and its containers assigned and allocated by DB2 based on those storage paths.

A database can only be enabled for automatic storage when it is first created. You cannot enable automatic storage for a database that was not originally defined to use it; Similarly, you cannot disable automatic storage for a database that was originally designed to use it.

DB2 creates an automatic storage database by default. The command line processor (CLP) provides a way to disable automatic storage during database creation by explicitly using the `AUTOMATIC STORAGE NO` clause.

The following are some examples of automatic storage being disabled explicitly:

```
CREATE DATABASE ASNOB1 AUTOMATIC STORAGE NO
CREATE DATABASE ASNOB2 AUTOMATIC STORAGE NO ON X:
```

The following are some examples of automatic storage being enabled either explicitly or implicitly:

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

Based on the syntax used, the DB2 database manager extracts the following two pieces of information that pertain to storage locations:

- The **database path** (which is where DB2 stores various control files for the database)
 - If the `DBPATH ON` clause is specified, this clause indicates the database path.
 - If the `DBPATH ON` clause is not specified, the first path listed in the `ON` clause indicates the database path (in addition to it being a storage path).
 - If neither the `DBPATH ON` nor the `ON` clauses are specified, the *dftdbpath* database manager configuration parameter is used to determine the database path.
- The **storage paths** (where DB2 creates automatic storage table space containers)
 - If the `ON` clause is specified, all of the listed paths are storage paths.
 - If the `ON` clause is not specified, there will be a single storage path that is set to the value of the *dftdbpath* database manager configuration parameter.

For the examples shown previously, the following table summarizes the storage paths used:

Table 3. Automatic storage database and storage paths.

CREATE DATABASE Command	Database Path	Storage Paths
CREATE DATABASE DB1 AUTOMATIC STORAGE YES	<dftdbpath>	<dftdbpath>
CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:	X:	X:
CREATE DATABASE DB3 ON /data/path1, /data/path2	/data/path1	/data/path1, /data/path2
CREATE DATABASE DB4 ON D:\StoragePath DBPATH ON C:	C:	D:\StoragePath

The storage paths provided must exist and be accessible. In a partitioned database environment, the same storage paths will be used on each database partition and they must exist and be accessible on each of those database partitions. There is no way to specify a unique set of storage paths for a particular database partition unless database partition expressions are used as part of the storage path name.

Doing this allows the database partition number to be reflected in the storage path such that the resulting path name is different on each database partition.

You use the argument

" \$N" ([blank]\$N)

to indicate a database partition expression. A database partition expression can be used anywhere in the storage path, and multiple database partition expressions can be specified. Terminate the database partition expression with a space character; whatever follows the space is appended to the storage path after the database partition expression is evaluated. If there is no space character in the storage path after the database partition expression, it is assumed that the rest of the string is part of the expression. The argument can only be used in one of the following forms:

Table 4. Automatic storage database syntax, examples and values.

Syntax	Example	Value
Operators are evaluated from left to right. % represents the modulus operator. The database partition number in the examples is assumed to be 10.		
[blank]\$N	" \$N"	10
[blank]\$N+[number]	" \$N+100"	110
[blank]\$N%[number]	" \$N%5"	0
[blank]\$N+[number]%[number]	" \$N+1%5"	1
[blank]\$N%[number]+[number]	" \$N%4+2"	4

For example:

```
CREATE DATABASE TESTDB ON "/path1ForNode $N", "/path2ForNode $N"
```

When free space is calculated for a storage path for a given database partition, the database manager will check for the existence of the following directories or mount points within the storage path and will use the first one that is found:

```
<storage path>/<instance name>NODE####/<database name>
<storage path>/<instance name>NODE####
<storage path>/<instance name>
<storage path>
```

Where:

<storage path> is a storage path associated with the database.
 <instance name> is the instance under which the database resides.
 NODE#### corresponds to the database partition number,
 for example:
 NODE0000 or NODE0001).
 <database name> is the name of the database.

In doing this, file systems can be mounted at a point beneath the storage path and the database manager will recognize that the actual amount of free space available for table space containers might not be the same amount that is associated with the storage path directory itself.

Consider the example where two logical database partitions exist on one physical computer and there is a single storage path: /db2data

Each database partition will use this storage path but the user might want to isolate the data from each database partition within its own file system. In this case, a separate file system can be created for each database partition and be mounted at /db2data/<instance>/NODE####

When creating containers on the storage path and determining free space, the database manager will know not to retrieve free space information for /db2data, but instead retrieve it for the corresponding /db2data/<instance>/NODE#### directory.

There are three default table spaces created whenever a database is created. If there are no explicit table space definitions provided as part of the CREATE DATABASE command, the table spaces are created as automatic storage table spaces.

After the database has been created, new storage paths can be added to the database using the ADD STORAGE clause of the ALTER DATABASE statement. For example:

```
ALTER DATABASE ADD STORAGE /data/path3, /data/path4
```

Related concepts:

- “Automatic storage table spaces” on page 58
- “How containers are added and extended in DMS table spaces” in *Administration Guide: Planning*
- “Table space maps” in *Administration Guide: Planning*

Related tasks:

- “Adding an automatic storage path” on page 64

Related reference:

- “Restore database implications” on page 59
- “Restrictions when using automatic storage” on page 62
- “ALTER DATABASE PARTITION GROUP statement” in *SQL Reference, Volume 2*
- “Monitoring storage paths” on page 62
- “ADD DBPARTITIONNUM command” in *Command Reference*
- “CREATE DATABASE command” in *Command Reference*
- “RESTORE DATABASE command” in *Command Reference*

Temporary automatic storage table spaces

This type of table space is created as SMS and all of the rules and behaviors associated with SMS table spaces still apply. However, there are differences with respect to how storage is managed shown in the following table:

Table 5. Differences managing non-automatic storage and automatic storage

Non-automatic storage	Automatic storage
Containers must be explicitly provided when the table space is created.	Containers cannot be provided when the table space is created, they are assigned and allocated automatically by the DB2 database manager.
A redirected restore operation can be used to redefine the containers associated with the table space.	A redirected restore operation cannot be used to redefine the containers associated with the table space because the DB2 database manager is in control of space management.

When a temporary automatic storage table space is created, the DB2 database manager examines all of the storage paths associated with the database and chooses the paths on which to create directory containers. Not all paths might be chosen because some might not have any free space or they are close to running out. Because the characteristics of the storage paths change over time, DB2 automatically redefines the containers whenever the database is started.

Related concepts:

- “Automatic storage databases” on page 54

Automatic storage table spaces

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

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

Here are some example statements that create automatic storage table spaces:

```
CREATE TABLESPACE TS1
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
CREATE TEMPORARY TABLESPACE TEMPTS
CREATE USER TEMPORARY TABLESPACE USRTMP MANAGED BY AUTOMATIC STORAGE
CREATE LONG TABLESPACE LONGTS
```

Although automatic storage table spaces appear to be a different table space type, it is really just an extension of the existing SMS and DMS types. If the table space being created is a `REGULAR` or `LARGE` table space, it is created as a DMS with file containers. If the table space being created is a `USER` or `SYSTEM TEMPORARY` table space, it is created as a SMS with directory containers.

Note: This behavior might change in future versions of the DB2 database manager.

The names associated with these containers have the following format:

```
<storage path>/<instance>/NODE####
/T#####
/C#####.<EXT>
```

where:

<storage path>	A storage path associated with the database
<instance>	The instance under which the database was created
NODE####	The database partition number (NODE0000 for example)
T#####	The table space ID (for example, T0000003)
C#####	The container ID (for example, C0000012)
<EXT>	An extension based on the type of data being stored:

CAT	System catalog table space
TMP	System temporary table space
UTM	User temporary table space
USR	User or regular table space
LRG	Large table space

Related concepts:

- “Automatic storage databases” on page 54
- “Temporary automatic storage table spaces” on page 57

Related tasks:

- “Viewing health alert objects” on page 447

Related reference:

- “Regular and large automatic storage table spaces” on page 63
- “Restrictions when using automatic storage” on page 62

Restore database implications

The RESTORE DATABASE command is used to restore a database from a backup image. During a restore it is possible to choose the location of the database path and its also possible to redefine the storage paths that are associated with the database. The database path and the storage paths are set using a combination of the TO, ON, and DBPATH ON clauses. For example, here are some valid RESTORE commands for databases enabled for automatic storage follow:

```
RESTORE DATABASE TEST1
RESTORE DATABASE TEST2 TO X:
RESTORE DATABASE TEST3 DBPATH ON D:
RESTORE DATABASE TEST3 ON /path1, /path2, /path3
RESTORE DATABASE TEST4 ON E:\newpath1, F:\newpath2 DBPATH ON D:
```

Like the CREATE DATABASE command, the DB2 database manager extracts the following two pieces of information that pertain to storage locations:

- The **database path** (which is where the DB2 database manager stores various control files for the database)
 - If the TO clause or the DBPATH ON clause is specified, the clause indicates the database path.
 - If the ON clause is used but the DBPATH ON clause is not specified with it, the first path listed in the ON clause is used as the database path (in addition to it being a storage path).
 - If none of the TO, ON, or DBPATH ON clauses are specified, the *dftdbpath* database manager configuration parameter determines the database path.

Note: If a database with the same name exists on disk, the database path is ignored, and the database is placed into the same location as the existing database.

- The **storage paths** (where DB2 creates automatic storage table space containers)
 - If the ON clause is specified, all of the paths listed are considered storage paths, and these paths are used instead of the ones stored within the backup image.
 - If the ON clause is not specified, no change is made to the storage paths (the storage paths stored within the backup image are maintained).

To make this concept clearer, the same five RESTORE command examples presented above are shown in the following table with their corresponding storage paths:

Table 6. Restore implications regarding database and storage paths

RESTORE DATABASE Command	No database on disk exists with same name		Database exists on disk with same name	
	Database path	Storage paths	Database path	Storage paths
RESTORE DATABASE TEST1	<dfldbpath>	Uses storage paths defined in the backup image	Uses database path of existing database	Uses storage paths defined in the backup image
RESTORE DATABASE TEST2 TO X:	X:	Uses storage paths defined in the backup image	Uses database path of existing database	Uses storage paths defined in the backup image
RESTORE DATABASE TEST3 DBPATH ON /db2/databases	/db2/databases	Uses storage paths defined in the backup image	Uses database path of existing database	Uses storage paths defined in the backup image
RESTORE DATABASE TEST4 ON /path1, /path2, /path3	/path1	/path1, /path2, /path3	Uses database path of existing database	/path1, /path2, /path3
RESTORE DATABASE TEST5 ON E:\newpath1, F:\newpath2 DBPATH ON D:	D:	E:\newpath1, F:\newpath2	Uses database path of existing database	E:\newpath1, F:\newpath2

For those cases where storage paths have been redefined as part of the restore operation, the table spaces that are defined to use automatic storage are automatically redirected to the new paths. However, you cannot explicitly redirect containers associated with automatic storage table spaces using the SET TABLESPACE CONTAINERS command; this action is not permitted.

Use the -s option of the *db2ckbkp* command to show whether or not automatic storage is enabled for a database within a backup image. The storage paths associated with the database are displayed if automatic storage is enabled.

For multi-partition automatic storage enabled databases, the **RESTORE DATABASE** command has a few extra implications:

1. The database must use the same set of storage paths on all database partitions.
2. Issuing a restore command with new storage paths can only be done on the catalog database partition, which will set the state of the database to RESTORE_PENDING on all non-catalog database partitions.

Table 7. Restore implications for multi-partition databases

RESTORE DATABASE Command	Issued on database partition #	No database on disk exists with same name		Database exists on disk with same name (includes skeleton databases)	
		Result on other database partitions	Storage paths	Result on other database partitions	Storage paths
RESTORE DATABASE TEST1	Catalog database partition	A skeleton database is created using the storage paths from the backup image on the catalog database partition. All other database partitions are placed in a RESTORE_PENDING state.	Uses storage paths defined in the backup image	Nothing. Storage paths have not changed so nothing happens to other database partitions	Uses storage paths defined in the backup image
	Non-catalog database partition	SQL#### is returned. If no database exists, the catalog database partition must be restored first.	N/A	Nothing. Storage paths have not changed so nothing happens to other database partitions	Uses storage paths defined in the backup image
RESTORE DATABASE TEST2 ON /path1, /path2, /path3	Catalog database partition	A skeleton database is created using the storage paths specified in the restore command. All other database partitions are placed in a RESTORE_PENDING state.	/path1, /path2, /path3		/path1, /path2, /path3
	Non-catalog database partition	SQL#### is returned. If no database exists, the catalog database partition must be restored first. Storage paths cannot be specified on the RESTORE of a non-catalog database partition.	N/A	SQL#### is returned. New storage paths cannot be specified on the RESTORE of a non-catalog database partition.	N/A

Related concepts:

- “Automatic storage databases” on page 54
- “Automatic storage table spaces” on page 58

Related tasks:

- “Adding an automatic storage path” on page 64

Related reference:

- “Regular and large automatic storage table spaces” on page 63
- “Restrictions when using automatic storage” on page 62

Monitoring storage paths

A database snapshot includes the list of storage paths associated with the database. If the number of automatic storage paths is 0, automatic storage is not enabled for the database.

```

Number of automatic storage paths      = ##
Automatic storage path                 = <1st path>
Automatic storage path                 = <2nd path>
...

```

If the bufferpool monitor switch is on, the following elements are also set:

```

File system ID                         = 12345
File system free space (bytes)         = 200000000000
File system used space (bytes)         = 40000000000000
File system total space (bytes)        = 40020000000000

```

This data is displayed on a per path basis: on a single database partition system per path, and per each database partition on a multi-database partitioned environment.

In addition, the following information is displayed within a table space snapshot. The information indicates whether or not the table space was created as an automatic storage table space.

```

Using automatic storage                 = Yes or No

```

Related concepts:

- “Automatic storage table spaces” on page 58
- “Automatic storage databases” on page 54
- “Temporary automatic storage table spaces” on page 57

Related reference:

- “Restore database implications” on page 59
- “Regular and large automatic storage table spaces” on page 63
- “Restrictions when using automatic storage” on page 62

Restrictions when using automatic storage

When deciding whether or not to create a database using automatic storage, the following restrictions are very important:

- There is no way to disable or enable automatic storage for a database after it has been created.

- Storage paths must be absolute path names. They can be paths or drive letters on Windows, but the database path must be a drive letter. The maximum path length is 175 characters.
- For partitioned databases, the same set of storage paths must be used on each database partition (unless database partition expressions are used).

Related concepts:

- “Automatic storage table spaces” on page 58
- “Automatic storage databases” on page 54
- “Temporary automatic storage table spaces” on page 57

Related reference:

- “Regular and large automatic storage table spaces” on page 63

Regular and large automatic storage table spaces

This type of table space is actually created as a DMS and all of the rules and behaviors associated with DMS table spaces still apply. However, there are differences with respect to how storage is managed shown in the following table:

Table 8. Differences managing non-automatic storage and automatic storage

Non-automatic storage	Automatic storage
Containers must be explicitly provided when the table space is created.	Containers cannot be provided when the table space is created, they will be assigned and allocated automatically by the DB2 database manager.
Automatic resizing of table spaces is off (AUTORESIZE NO) by default.	Automatic resizing of table spaces is on (AUTORESIZE YES) by default.
The initial size for the table space cannot be specified using the INITIALSIZE clause.	The initial size for the table space can be specified using the INITIALSIZE clause.
Container operations can be performed using the ALTER TABLESPACE statement (ADD, DROP, BEGIN NEW STRIPE SET, and so on).	Container operations cannot be performed because the DB2 database manager is in control of space management.
A redirected restore operation can be used to redefine the containers associated with the table space.	A redirected restore operation cannot be used to redefine the containers associated with the table space because the DB2 database manager is in control of space management.

When a regular or large automatic storage table space is created, an initial size can be specified as part of the CREATE TABLESPACE statement. For example:

```
CREATE TABLESPACE TS1 INITIALSIZE 100 M
```

If the initial size isn't specified then DB2 will use a default value of 32 megabytes.

To create a table space with a given size, the DB2 database manager creates file containers within the storage paths. If there is an uneven distribution of space among the paths, containers might be created with different sizes. As a result, it is important that all of the storage paths have a similar amount of free space on them.

If automatic resizing is enabled for the table space, as space is used within it, the DB2 database manager automatically extends existing containers and adds new ones (using stripe sets). Whether containers are extended or added, no rebalance ever takes place.

Related concepts:

- “Automatic storage databases” on page 54
- “Temporary automatic storage table spaces” on page 57
- “Automatic storage table spaces” on page 58

Related tasks:

- “Viewing health alert objects” on page 447

Related reference:

- “Monitoring storage paths” on page 62
- “Restore database implications” on page 59
- “Restrictions when using automatic storage” on page 62

Adding an automatic storage path

You can add a storage path to a database that is enabled for automatic storage.

When you add a storage path for a multiple-partition database environment, the exact storage path must be replicated on each database partition. A path and its associated folders must be created on each database partition. For this reason, the new folder icon is unavailable when adding a storage path. If a specified path does not exist on every database partition, the statement is rolled back.

Restrictions:

A database is enabled for automatic storage when it is created. You cannot enable automatic storage for a database that was not originally defined as an automatic storage database.

Procedure:

To add a storage path to an existing database using the Control Center:

1. Expand the object tree until you see the **Table Spaces** folder of the database to which you want to add a storage path. Right-click the **Table Spaces** folder and select **Manage Storage**→**Add Automatic Storage** from the pop-up menu. The Add Storage window opens.
2. Click **Add**. The Add Storage Path window opens.
3. Specify the storage path.

To add a storage path to an existing database using the command line, use the **ALTER DATABASE** statement.

Related concepts:

- “Automatic storage databases” on page 54

Related reference:

- “ALTER DATABASE statement” in *SQL Reference, Volume 2*

License management

The management of licenses for your DB2 products is done primarily through the License Center within the Control Center of the online interface to the product.

From the License Center you can check the license information, statistics, authorized 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 concepts:

- “Control Center overview” on page 376
- “License Center overview” on page 411

Related reference:

- “db2licm - License management tool command” in *Command Reference*

Registry and environment variables

This section introduces registry and environment variables. More detailed information is provided in Part 2 of this manual.

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 database profile registry, changing your environment variables on Windows workstations (for example) required you to change an environment variable and restart. Now, your environment is controlled, with a few exceptions, by registry variables stored in the DB2 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 restarting; this information is stored immediately in the profile registries. The DB2 registry applies the updated information to DB2 server instances and DB2 applications started after the changes are made.

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

Note: There are DB2 environment variables DB2INSTANCE, and DB2NODE which might not be stored in the DB2 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 restarted. On UNIX platforms, the **export** command might 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 Instance Level Profile Registry. The majority of the DB2 environment variables are placed within this registry. The environment variable settings for a particular instance are kept in this registry. Values defined in this level override their settings in the global level.
- The DB2 Global Level Profile Registry. If an environment variable is not set for a particular instance, this registry is used. This registry is visible to all instances pertaining to a particular copy of DB2 ESE, one global-level profile exists in the installation path.
- The DB2 Instance Node Level Profile Registry. This registry level contains variable settings that are specific to a database partition in a partitioned database environment. Values defined in this level override their settings at the instance and global levels.
- The DB2 Instance Profile Registry. This registry contains a list of all instance names associated with the current copy. Each installation has its own list. You can see the complete list of all the instances available on the system by running `db2ilist`.

DB2 configures the operating environment by checking for registry values and environment variables and resolving them in the following order:

1. Environment variables set with the `set` command. (Or the `export` command on UNIX platforms.)
2. Registry values set with the instance node level profile (using the `db2set -i <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 database partitions that are identified as “red”, “white”, and “blue”. On UNIX platforms, if the instance owner runs the following from any of the database partitions:

```
db2set -i F00=BAR
```

or

```
db2set F00=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 `sql1ib` directory. In partitioned database environments, the `sql1ib` directory is located on the filesystem shared by all physical database partitions.

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. The DB2 database manager stores the instance level profile registry inside the Windows registry. There is no sharing across physical database partitions. To set the registry variables on all the physical computers, use the “`rah`” command as follows:

```
rah db2set -i F00=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 computers are configured to refer to those on the instance owning computer. This effectively creates an environment where the registry variables on the instance-owning computer are shared amongst all computers in the instance.

Using the example shown above, and assuming that “red” is the owning computer, then one would set DB2REMOTEPREG on “white” and “blue” computers 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 the DB2 database manager reads the registry variables on Windows, it first reads the DB2REMOTEPREG value. If DB2REMOTEPREG is set, it then opens the registry on the remote computer whose computer name is specified in the DB2REMOTEPREG variable. Subsequent reading and updating of the registry variables will be redirected to the specified remote computer.

Accessing the remote registry requires that the Remote Registry Service is running on the target computer. Also, the user logon account and all DB2 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 computers belong to the same MSCS cluster, the registry variables are maintained in the cluster registry. Therefore, they are already shared between all computers 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 database partitions span across multiple MSCS clusters, you cannot use DB2REMOTEPREG to point to the instance-owning computer because the registry variables of the instance-owning computer reside in the cluster registry.

Related concepts:

- “DB2 registry and environment variables” in *Performance Guide*

Related tasks:

- “Declaring, showing, changing, resetting, and deleting registry and environment variables” on page 68

Setting the current instance environment variables

Procedure:

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

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

```
set db2instance=<new_instance_name>
```
- If the DB2INSTANCE environment variable is not set for the current session, the DB2 database manager uses the setting for the DB2INSTANCE environment variable from the system environment variables. On Windows, system environment variables are set in the System Environment registry.
- If the DB2INSTANCE environment variable is not set at all, the DB2 database manager 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, showing, changing, resetting, and deleting registry and environment variables” on page 68

Declaring, showing, changing, resetting, and deleting registry and environment variables

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

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

Procedure:

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 database partition in an instance, use:

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

To change a registry variable default for all instances pertaining to a particular installation in the system, use:

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

If you use an aggregate registry variable such as DB2_WORKLOAD to define your environment, you can set that variable using:

```
db2set DB2_WORKLOAD=SAP
```

If you use 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, you can set a DB2 registry variable value so that its scope is global to all servers and all users that belong to a directory partition or to a Windows domain. Currently, there are only two DB2 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 profile.env file is updated by two or more users with the db2set command at the same time, or very close to the same time, 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 registry variables at the computer global level, and the -gl option which is specifically used at 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 (global does not mean the variables will be shared between copies of DB2). They cannot be set at the instance or database partition 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 database partition in an instance back to the default found in the Global Profile Registry, use:

```
db2set -r registry_variable_name database_partition_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 database partition level, enter:

```
db2set registry_variable_name= -i instance_name  
database_partition_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 deletes the setting for the parameter you specify and restricts high level profiles from changing this variable's value (in this case, DB2 global-level profile). However, the variable you specify could still be set by a lower level profile (in this case, the DB2 database partition-level profile).

Related concepts:

- "DB2 registry and environment variables" in *Performance Guide*

Related tasks:

- "Searching the LDAP servers" on page 585
- "Setting DB2 registry variables at the user level in the LDAP environment" on page 587
- "Setting environment variables on UNIX systems" on page 79
- "Setting environment variables on Windows" on page 77

General registry variables

DB2ACCOUNT

- Operating system: All
- Default=null
- This variable defines the accounting string that is sent to the remote host. Refer to the *DB2 Connect User's Guide* for details.

DB2BIDI

- Operating system: All
- Default=NO, Values: YES or NO
- This variable enables bidirectional support and the DB2CODEPAGE variable is used to declare the code page to be used.

DB2CODEPAGE

- Operating system: All

- Default: derived from the language ID, as specified by the operating system.
- This variable specifies the code page of the data presented to DB2 for database client application. The user should not set DB2CODEPAGE unless explicitly stated in DB2 documents, or asked to do so by DB2 service. Setting DB2CODEPAGE to a value not supported by the operating system can produce unexpected results. Normally, you do not need to set DB2CODEPAGE because DB2 automatically derives the code page information from the operating system.

Note: Because Windows does not report a Unicode code page (in the Windows regional settings) instead of the ANSI code page, a Windows application will not behave as a Unicode client. To override this behavior, set the DB2CODEPAGE registry variable to 1208 (for the Unicode code page) to cause the application to behave as a Unicode application.

DB2_COLLECT_TS_REC_INFO

- Operating system: All
- Default=ON, Values: ON or OFF
- This variable specifies whether DB2 will process all log files when rolling forward a table space, regardless of whether the log files contain log records that affect the table space. To skip the log files known not to contain any log records affecting the table space, set this variable to "ON". DB2_COLLECT_TS_REC_INFO must be set before the log files are created and used so that the information required for skipping log files is collected.

DB2_CONNRETRIES_INTERVAL

- Operating system: All
- Default= not set, Values: an integer number of seconds
- This variable specifies the sleep time between consecutive connection retries, in seconds, for the automatic client reroute feature. You can use this variable in conjunction with DB2_MAX_CLIENT_CONNRETRIES to configure the retry behavior for automatic client reroute.

If DB2_MAX_CLIENT_CONNRETRIES is set, but DB2_CONNRETRIES_INTERVAL is not, DB2_CONNRETRIES_INTERVAL defaults to 30. If DB2_MAX_CLIENT_CONNRETRIES is not set, but DB2_CONNRETRIES_INTERVAL is set, DB2_MAX_CLIENT_CONNRETRIES defaults to 10. If neither DB2_MAX_CLIENT_CONNRETRIES nor DB2_CONNRETRIES_INTERVAL is set, the automatic client reroute feature reverts to its default behavior of retrying the connection to a database repeatedly for up to 10 minutes.

DB2CONSOLECP

- Operating system: Windows
- Default= null, Values: all valid code page values
- Specifies the codepage for displaying DB2 message text. When specified, this value overrides the operating system codepage setting.

DB2COUNTRY

- Operating system: Windows
- Default=null, Values: all valid numeric country, territory, or region codes

- This variable specifies the country, territory, or region code of the client application. When specified, this value overrides the operating system setting.

DB2DBDFT

- Operating system: All
- Default=null
- This variable specifies the database alias name of the database to be used for implicit connects. If an application has no database connection but SQL or XQuery statements are issued, an implicit connect will be made if the DB2DBDFT environment variable has been defined with a default database.

DB2DBMSADDR

- Operating system: Windows 32-bit
- Default= 0x20000000, Values: 0x20000000 to 0xB0000000 in increments of 0x10000
- This variable specifies the default database manager shared memory address in hexadecimal format. If *db2start* fails due to a shared memory address collision, this registry variable can be modified to force the database manager instance to allocate its shared memory at a different address.

DB2DISCOVERYTIME

- Operating system: Windows
- Default=40 seconds, Minimum=20 seconds
- This variable specifies the amount of time that SEARCH discovery will search for DB2 systems.

DB2FFDC

- Operating system: All
- Default: ON, Values: ON, CORE:OFF
- Provides the ability to deactivate core file generation. By default, this registry variable is set to ON. If this registry variable is not set, or is set to a value other than CORE:OFF, core files may be generated if the DB2 server abends. (Core files are used for problem determination, and are created in the DIAGPATH.)

Note: On Linux platforms, the default core file size limit is set to 0 (that is, `ulimit -c`). With this setting, core files are not generated. To allow core files to be created on Linux platforms, set the value to unlimited.

Core files contain the entire process image of the terminating DB2 process. Consideration should be given to the available file system space as core files can be quite large. The size is dependent on the DB2 configuration and the state of the process at the time the problem occurs.

DB2_FORCE_APP_ON_MAX_LOG

- Operating system: All
- Default: TRUE, Values: TRUE, FALSE
- Specifies what happens when the MAX_LOG configuration parameter value is exceeded. If set to TRUE, the application is forced off the database and the unit of work is rolled back.

If FALSE, the current statement fails. The application can still commit the work completed by previous statements in the unit of work, or it can roll back the work completed to undo the unit of work.

DB2GRAPHICUNICODESERVER

- Operating system: All
- Default=OFF, Values: ON or OFF
- This registry variable is used to accommodate existing applications written to insert graphic data into a Unicode database. Its use is only needed for applications that specifically send sqldbcchar (graphic) data in Unicode instead of the code page of the client. (sqldbcchar is a supported SQL data type in C and C++ that can hold a single double-byte character.) When set to "ON", you are telling the database that graphic data is coming in Unicode, and the application expects to receive graphic data in Unicode.

DB2INCLUDE

- Operating system: All
- Default=current directory
- Specifies a path to be used during the processing of the SQL INCLUDE text-file statement during DB2 PREP processing. It provides a list of directories where the INCLUDE file might be found. Refer to *Developing Embedded SQL Applications* for descriptions of how DB2INCLUDE is used in the different precompiled languages.

DB2INSTDEF

- Operating system: Windows
- Default=DB2
- This variable sets the value to be used if DB2INSTANCE is not defined.

DB2INSTOWNER

- Operating system: Windows
- Default=null
- The registry variable created in the DB2 profile registry when the instance is first created. This variable is set to the name of the instance-owning machine.

DB2_LIC_STAT_SIZE

- Operating system: All
- Default=null, Range: 0 to 32 767
- This variable determines the maximum size (in MBs) of the file containing the license statistics for the system. A value of zero turns the license statistic gathering off. If the variable is not recognized or not defined, the variable defaults to unlimited. The statistics are displayed using the License Center.

DB2LOCALE

- Operating system: All
- Default= NO, Values: YES or NO
- This variable specifies whether the default "C" locale of a process is restored to the default "C" locale after calling DB2 and whether to restore the process locale back to the original 'C' after calling a DB2 function. If the original locale was not 'C', then this registry variable is ignored.

DB2_MAX_CLIENT_CONNRETRIES

- Operating system: All
- Default=not set, Values: an integer number of maximum times to retry the connection
- This variable specifies the maximum number of connection retries that the automatic client reroute feature will attempt. You can use this variable in conjunction with DB2_CONNRETRIES_INTERVAL to configure the retry behavior for automatic client reroute.

If DB2_MAX_CLIENT_CONNRETRIES is set, but DB2_CONNRETRIES_INTERVAL is not, DB2_CONNRETRIES_INTERVAL defaults to 30. If DB2_MAX_CLIENT_CONNRETRIES is not set, but DB2_CONNRETRIES_INTERVAL is set, DB2_MAX_CLIENT_CONNRETRIES defaults to 10. If neither DB2_MAX_CLIENT_CONNRETRIES nor DB2_CONNRETRIES_INTERVAL is set, the automatic client reroute feature reverts to its default behavior of retrying the connection to a database repeatedly for up to 10 minutes.

DB2NBDISCOVERRCVBUFS

- Operating system: All
- Default=16 buffers, Minimum=16 buffers
- This variable is used for NetBIOS search discovery. The variable specifies the number of concurrent discovery responses that can be received by a client. If the client receives more concurrent responses than are specified by this variable, then the excess responses are discarded by the NetBIOS layer. The default is sixteen (16) NetBIOS receive buffers. If a number less than the default value is chosen, then the default is used.

DB2_OBJECT_TABLE_ENTRIES

- Operating system: All
 - Default=0, Values: 0–65532
- The actual maximum value possible on your system depends on the page size and extent size, but it cannot exceed 65532.
- This variable specifies the expected number of objects in a table space. If you know that a large number of objects (for example, 1000 or more) will be created in a DMS table space, you should set this registry variable to the approximate number before creating the table space. This will reserve contiguous storage for object metadata during table space creation. Reserving contiguous storage reduces the chance that an online backup will block operations which update entries in the metadata (for example, CREATE INDEX, IMPORT REPLACE). It will also make resizing the table space easier because the metadata will be stored at the start of the table space.

If the initial size of the table space is not large enough to reserve the contiguous storage, the table space creation will continue without the additional space reserved.

DB2TERRITORY

- Operating system: All
- Default: derived from the language ID, as specified by the operating system.
- This variable specifies the region, or territory code of the client application, which influences date and time formats.

DB2_VIEW_REOPT_VALUES

- Operating system: All
- Default=NO, Values: YES, NO
- This variable enables all users to store the cached values of a reoptimized SQL or XQuery statement in the EXPLAIN_PREDICATE table when the statement is explained. When this variable is set to NO, only DBADM is allowed to save these values in the EXPLAIN_PREDICATE table.

Related concepts:

- “DB2 registry and environment variables” in *Performance Guide*

Aggregate registry variables

An aggregate registry variable allows several registry variables to be grouped as a configuration that is identified by another registry variable name. Each registry variable that is part of the group has a predefined setting. The aggregate registry variable is given a value that is interpreted as declaring several registry variables.

The intention of an aggregate registry variable is to ease registry configuration for broad operational objectives.

The only valid aggregate registry variable is DB2_WORKLOAD.

The only valid value for this variable is SAP.

When you have set DB2_WORKLOAD=SAP, the user table space SYSTOOLSPACE and the user temporary table space SYSTOOLSTMPSPACE are not automatically created. These table spaces are used for tables created automatically by the following wizards, utilities, or functions:

- Automatic maintenance
- Design advisor
- Control Center database information panel
- SYSINSTALLOBJECTS stored procedure, if the table space input parameter is not specified
- GET_DBSIZE_INFO stored procedure

Without the SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces, you cannot use these wizards, utilities, or functions.

To be able to use these wizards, utilities, or functions, do either of the following:

- Manually create SYSTOOLSPACE (on the catalog node only, if using the Database Partition Feature (DPF). For example:

```
CREATE REGULAR TABLESPACE SYSTOOLSPACE
  IN IBMCATGROUP
  MANAGED BY SYSTEM
  USING ('SYSTOOLSPACE')
```

- Call SYSINSTALLOBJECTS to create these objects, specifying a valid table space, for each of the following tool names: “DB2AC”, “POLICY”, and “STMG_DBSIZE_INFO”

After completing at least one of these choices, create a user temporary table space (also on the catalog node only, if using the Database Partition Feature (DPF). For example:

```
CREATE USER TEMPORARY TABLESPACE SYSTOOLSTMPSPACE
  IN IBMCATGROUP
  MANAGED BY SYSTEM
  USING ('SYSTOOLSTMPSPACE')
```

Once the table space SYSTOOLSPACE and the temporary table space SYSTOOLSTMPSPACE are created, you can use the wizards, utilities, or functions mentioned earlier.

Any registry variable that is implicitly configured through an aggregate registry variable might also be explicitly defined. Explicitly setting a registry variable that was previously given a value through the use of an aggregate registry variable is useful when doing performance or diagnostic testing. Explicitly setting a variable implicitly that is configured by an aggregate is referred to as “overriding” the variable.

When you explicitly set a registry variable which is then overridden by using an aggregate registry variable, a warning is issued. This warning tells you that the explicit value is maintained. If the aggregate registry variable is used first and then you specify an explicit registry variable, a warning is not given.

None of the registry variables that are configured through setting an aggregate registry variable are shown unless you explicitly make that request for each variable. When you query the aggregate registry variable, only the value assigned to that variable is shown. Most of your users should not care about the values for each individual variable.

The following example shows the interaction between using the aggregate registry variable and explicitly setting a registry variable. For example, you might have set the DB2_WORKLOAD aggregate registry variable to SAP and have overridden the DB2_SKIPDELETED registry variable to NO. By entering db2set, you would receive the following results:

```
DB2_WORKLOAD=SAP
DB2_SKIPDELETED=NO
```

In another situation, you might have set DB2ENVLIST, set the DB2_WORKLOAD aggregate registry variable to SAP, and overridden the DB2_SKIPDELETED registry variable to NO. (This assumes that the DB2_SKIPDELETED registry variable is part of the group making up the SAP environment.) In addition, those registry variables that were configured automatically through setting the aggregate registry variable will show the name of the aggregate displayed within square brackets, adjacent to its value. The DB2_SKIPDELETED registry variable will show a “NO” value and will show “[O]” displayed adjacent to its value.

When you no longer require the configuration associated with DB2_WORKLOAD, you can disable the implicit values of each registry variable in the group by deleting the aggregate registry variable’s value. After deleting the DB2_WORKLOAD aggregate registry variable’s value and restarting DB2, DB2 behaves as if none of the registry variables implicitly configured as part of the SAP environment are in effect. The method used to delete an aggregate registry variable’s value is the same as deleting an individual registry variable.

To disable SAP environment support, use:

```
db2set DB2_WORKLOAD=
```

Deleting an aggregate registry variable's value does not delete a registry variable's value that has been explicitly set. It does not matter that the registry variable is a member of the group definition being disabled. The explicit setting for the registry variable is maintained.

You might need to see the values for each registry variable that is a member of the DB2_WORKLOAD aggregate registry variable. Before setting the DB2_WORKLOAD aggregate registry variable to SAP, and assuming that no registry variables that are included in the group are explicitly defined, you might want to see the values that would be used if you configured the DB2_WORKLOAD aggregate registry variable to SAP. To find the values that would be used if DB2_WORKLOAD=SAP, run `db2set -gd DB2_WORKLOAD=SAP`. This returns a list of registry variables and their values.

Related concepts:

- “Environment variables and the profile registry” on page 65
- “DB2 registry and environment variables” in *Performance Guide*

Related tasks:

- “Declaring, showing, changing, resetting, and deleting registry and environment variables” on page 68
- “Setting DB2 registry variables at the user level in the LDAP environment” on page 587

Related reference:

- “General registry variables” on page 70

Setting environment variables on Windows

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 profile registry variable DB2INSTDEF might be set in the global level profile to specify the instance name to use if DB2INSTANCE is not defined.

DB2 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 profile registry variable DB2INSTDEF might 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 computer. This environment variable must be set in the session in which the application or command is issued and not in the DB2 profile registry. If this variable is not set, the target logical node defaults to the logical node which is defined as zero (0) on the computer.

Procedure:

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%
```

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

- Select **Start**→**Control Panel**.
- Depending on the Windows theme and the currently selected view type, you might have to select **Performance** and **Maintenance** before you can select the **System** icon.
- From the System Properties window, select the **Advanced** tab; click **Environment Variables** and do the following:
 1. If the DB2INSTANCE variable does not exist:
 - a. Click **New**.
 - b. Fill in the *Variable Name* field with DB2INSTANCE.
 - c. 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. Restart your system for these changes to take effect.

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

```
set DB2INSTANCE=TEST
db2start
```

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 Instance Level Profile Registry in the Windows operating system registry, with the path:

```
\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES\instance_name
```

Note: The *instance_name* is the name of the DB2 instance.

- The DB2 Global Level Profile Registry in the Windows registry, with the path:

```
\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\GLOBAL_PROFILE
```

- The DB2 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 Instance Profile Registry required. For each of the DB2 instances in the system, a key is created in the path:

```
\HKEY_LOCAL_computer\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 91

Related tasks:

- “Setting environment variables on UNIX systems” on page 79

Setting environment variables on UNIX systems

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 profile registry. To set variables that are not supported by the DB2 database manager, define them in your script files, userprofile and usercshrc.

An instance owner or SYSADM user might 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.

Procedure:

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 profile registry to be administered properly, the following file ownership rules must be followed on UNIX operating systems.

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

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

The access permissions and ownership of this file should be:

```
-rw-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 Global Level Profile Registry is located under:
 - <installation path>/default.env for all UNIX and LINUX platforms.

The access permissions and ownership of this file should be:

```
-rw-rw-r-- root <group> default.env
```

where <group> is the group name for group ID 0; for example, on AIX, it is "system".

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

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

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

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

```
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 Instance Profile Registry is located under:

– <installation path>/profiles.reg for all UNIX and LINUX platforms

The access permissions and ownership of this file should be:

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

Related concepts:

- "DB2 Administration Server" on page 91

Related tasks:

- "Setting environment variables on Windows" on page 77

Configuration files and parameters

This section describes configuration files and parameters.

Database configuration file

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 database 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 might 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 might need to be updated to achieve optimal performance for your database.

For multi-partition databases: When you have a database that is distributed across more than one database partition, the configuration file should be the same on all database partitions. Consistency is required since the query 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 130

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

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 `sql1ib` subdirectory of the home directory for the instance before you can start the database manager with parallel capabilities across multiple database 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 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 from an instance. You can use the **db2nchg** command to modify a database partition server configuration including moving the database partition server from one computer 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 `sql1ib` subdirectory other than those created by the DB2 database manager 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 `sql1ib` 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 database partition. 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 distribution map, which specifies how data is distributed, would be compromised.)

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

The database partition number is used to generate a database partition 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 might 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 database partition. 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 database partitions.

For each hostname, one *logical-port* must be either 0 (zero) or blank (which defaults to 0). The database partition 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.

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 database interface. It also shows the database partition numbers starting at 1 (rather than at 0), and a gap in the *dbpartitionnum* sequence:

Table 9. 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 database partitioning requires that 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 *SQL Guide*

Related reference:

- “CREATE DATABASE command” in *Command Reference*
- “db2nchg - Change database partition server configuration command” in *Command Reference*
- “db2ncrt - Add database partition server to an instance command” in *Command Reference*
- “db2ndrop - Drop database partition server from an instance command” in *Command Reference*
- “db2start - Start DB2 command” in *Command Reference*
- “db2stop - Stop DB2 command” in *Command Reference*
- “DROP DATABASE command” in *Command Reference*

Defining the scope of configuration parameters using the Configuration Advisor

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. In Version 9.1, the Configuration Advisor is automatically invoked when you create a database. To disable this feature, or to explicitly enable it, use the **db2set** command before creating the database. Examples:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

See Automatic features enabled by default for other DB2 features that are enabled by default.

Prerequisites:

The database is already created.

Procedure:

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.

Note: Even if the Configuration Advisor was automatically enabled for the CREATE DATABASE request, if desired, you can still specify AUTOCONFIGURE <options>, in particular for APPLY DB and APPLY DBM, in order to apply the database manager configuration recommendation values. If the Configuration Advisor was disabled for the CREATE DATABASE request, then you can run it manually afterwards with the given options.

Related concepts:

- “Automatic features enabled by default” in *Administration Guide: Planning*
- “Configuration parameters” in *Performance Guide*

Related tasks:

- “Creating a database” on page 113

Related reference:

- “AUTOCONFIGURE command” in *Command Reference*
- “AUTOCONFIGURE command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

Generating recommendations for database configuration

The Configuration Advisor is used to make recommendations for the initial values of the buffer pool size, database configuration parameters, and database manager configuration parameters. The recommended values can be displayed or applied by using the APPLY option. The recommendations are based on input that you provide and system information that the advisor gathers.

The values suggested by the Configuration Advisor are relevant for only one database per instance. If you want to use this advisor on more than one database, each database should belong to a separate instance.

In Version 9.1, the Configuration Advisor is automatically invoked when you create a database. To disable this feature for a given database, or to explicitly enable it, use the **db2set** command. Examples:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

Prerequisites:

You can use the AUTOCONFIGURE option on an existing database or as an option to the **CREATE DATABASE** command.

To configure your database you must have either SYSADM, SYSCTRL, or SYSMANT authority.

Procedure:

The Configuration Advisor can be run through the **AUTOCONFIGURE** command on the command line processor (CLP), through the Configuration Advisor GUI in the Control Center, or by calling the **db2AutoConfig** API.

To open the Configuration Advisor from the Control Center:

1. Expand the object tree until you find the database object for which you would like DB2 to provide configuration recommendations.
2. Right-click the database and select **Configure Advisor** from the pop-up menu. The Configuration Advisor opens.

Detailed information is provided through the online help facility within the Control Center.

To request configuration recommendations using the command line, enter:

```
AUTOCONFIGURE
  USING <input_keyword> <param_value>
  APPLY <value>
```

The following is an example of an **AUTOCONFIGURE** command that requests configuration recommendations based on input about how the database is used, but specifies that the recommendations should not be applied:

```
DB2 AUTOCONFIGURE USING
  MEM_PERCENT 60
  WORKLOAD_TYPE MIXED
  NUM_STMTS 500
  ADMIN_PRIORITY BOTH
  IS_POPULATED YES
  NUM_LOCAL_APPS 0
  NUM_REMOTE_APPS 20
  ISOLATION RR
  BP_RESIZEABLE YES
APPLY NONE
```

Related concepts:

- “Automatic features enabled by default” in *Administration Guide: Planning*

Related tasks:

- “Creating a database” on page 113

Related reference:

- “Configuration Advisor sample output” on page 85
- “AUTOCONFIGURE command” in *Command Reference*
- “db2AutoConfig API - Access the Configuration Advisor” in *Administrative API Reference*

Configuration Advisor sample output

To demonstrate how the Configuration Advisor works, here is an example of the command used to request configuration recommendations using the CLP, along with its corresponding output:

1. Connect to the database:
DB2 CONNECT TO PERSONL
2. Issue the **AUTOCONFIGURE** command, specifying input that indicates how the database is used. If you want to view the configuration recommendations, but not apply them, you can set a value of NONE for the APPLY option, as in the following example:

```
DB2 AUTOCONFIGURE USING
  MEM_PERCENT 60
  WORKLOAD_TYPE MIXED
  NUM_STMTS 500
  ADMIN_PRIORITY BOTH
  IS_POPULATED YES
  NUM_LOCAL_APPS 0
  NUM_REMOTE_APPS 20
  ISOLATION RR
  BP_RESIZEABLE YES
APPLY NONE
```

If you are unsure about a hint value, that is, the parameters passed to the command, you can omit it and the default will be used. When using the Advisor, you can pass up to 10 hints: MEM_PERCENT, WORKLOAD_TYPE,

and so on, as shown above. Each hint has a range of acceptable values; for example, 1-100 is valid for MEM_PERCENT. If the value for this parameter is omitted, its default 25 is used.

After the **AUTOCONFIGURE** command is issued from the CLP, the recommendations are displayed to the screen in table format.

Table 10. Configuration Advisor sample output: Part 1

Former and Applied Values for Database Manager Configuration			
Description	Parameter	Current Value	Recommended Value
Application support layer heap size (4KB)	(ASLHEAPSZ) = 15		15
No. of int. communication buffers(4KB)	(FCM_NUM_BUFFERS) = AUTOMATIC		AUTOMATIC
Enable intra-partition parallelism	(INTRA_PARALLEL) = NO		NO
Maximum query degree of parallelism	(MAX_QUERYDEGREE) = ANY		1
Max number of existing agents	(MAXAGENTS) = 200		200
Agent pool size	(NUM_POOLAGENTS) = 100(calculated)		200
Initial number of agents in pool	(NUM_INITAGENTS) = 0		0
Max requester I/O block size (bytes)	(RQRIOBLK) = 32767		32767
Sort heap threshold (4KB)	(SHEAPTHRES) = 0		0

Table 11. Configuration Advisor sample output: Part 2

Former and Applied Values for Database Configuration			
Description	Parameter	Current Value	Recommended Value
Max appl. control heap size (4KB)	(APP_CTL_HEAP_SZ) = 128		128
Max size of appl. group mem set (4KB)	(APPGROUP_MEM_SZ) = 20000		20000
Default application heap (4KB)	(APPLHEAPSZ) = 256		256
Catalog cache size (4KB)	(CATALOGCACHE_SZ) = (MAXAPPLS*4)		260
Changed pages threshold	(CHNGPGS_THRESH) = 60		80
Database heap (4KB)	(DBHEAP) = 1200		2791
Degree of parallelism	(DFT_DEGREE) = 1		1
Default tablespace extentsize (pages)	(DFT_EXTENT_SZ) = 32		32
Default prefetch size (pages)	(DFT_PREFETCH_SZ) = AUTOMATIC		AUTOMATIC
Default query optimization class	(DFT_QUERYOPT) = 5		5
Max storage for lock list (4KB)	(LOCKLIST) = 100		AUTOMATIC
Log buffer size (4KB)	(LOGBUFSZ) = 8		99
Log file size (4KB)	(LOGFILSIZ) = 1000		1024
Number of primary log files	(LOGPRIMARY) = 3		8
Number of secondary log files	(LOGSECOND) = 2		3
Max number of active applications	(MAXAPPLS) = AUTOMATIC		AUTOMATIC
Percent. of lock lists per application	(MAXLOCKS) = 10		AUTOMATIC
Group commit count	(MINCOMMIT) = 1		1
Number of asynchronous page cleaners	(NUM_IOCLEANERS) = 1		1
Number of I/O servers	(NUM_IOSERVERS) = 3		4
Package cache size (4KB)	(PCKCACHESZ) = (MAXAPPLS*8)		1533
Percent log file reclaimed before soft chckpt	(SOFTMAX) = 100		320
Sort list heap (4KB)	(SORTHEAP) = 256		AUTOMATIC
SQL statement heap (4KB)	(STMTHEAP) = 4096		4096
Statistics heap size (4KB)	(STAT_HEAP_SZ) = 4384		4384
Utilities heap size (4KB)	(UTIL_HEAP_SZ) = 5000		113661
Self tuning memory	(SELF_TUNING_MEM) = ON		ON
Automatic runstats	(AUTO_RUNSTATS) = ON		ON
Sort heap thres for shared sorts (4KB)	(SHEAPTHRES_SHR) = 5000		AUTOMATIC

Table 12. Configuration Advisor sample output: Part 3

Former and Applied Values for Bufferpool(s)			
Description	Parameter	Current Value	Recommended Value
IBMDEFAULTBP	Bufferpool size =	-2	340985

DB210203I AUTOCONFIGURE completed successfully. Database manager or database configuration values may have been changed. The instance must be restarted before any changes come into effect. You may also want to rebind your packages after the new configuration parameters take effect so that the new values will be used.

If you agree with all of the recommendations, you can reissue the **AUTOCONFIGURE** command, but specify that you want the recommended values to be applied. Otherwise, you can update individual configuration parameters using the **UPDATE DATABASE MANAGER CONFIGURATION** command and the **UPDATE DATABASE CONFIGURATION** command.

Note: Self tuning memory is enabled by default on database creation for single-partition databases only. For newly created multi-partitioned databases, this feature disabled by default.

Related tasks:

- “Generating recommendations for database configuration” on page 84

Related reference:

- “AUTOCONFIGURE command” in *Command Reference*
- “UPDATE DATABASE CONFIGURATION command” in *Command Reference*
- “UPDATE DATABASE MANAGER CONFIGURATION command” in *Command Reference*
- “db2AutoConfig API - Access the Configuration Advisor” in *Administrative API Reference*

Database history file

A recovery history file is created with each database and is automatically updated whenever a database or table space is created, altered, renamed, loaded, backed up, restored, rolled-back, quiesced, or dropped. The DB_HISTORY administrative view returns information from the history files from all database partitions.

Accessing the history file using the DB_HISTORY administrative view

You can use the DB_HISTORY() administrative view to access the contents of the database history file. This method is an alternative to using the LIST HISTORY CLP command or the C history APIs.

Deletes and updates to the database history file can only be done through the PRUNE or UPDATE HISTORY commands.

Prerequisites:

A database connection is required to use this function.

Restrictions:

This administrative view is not available for databases created using DB2 Universal Database™ Version 8.2 and earlier.

Procedure:

To access the history file:

1. Ensure you are connected to a database.
2. Use the DB_HISTORY() administrative view within an SQL SELECT statement to access the database history file for the database you are connected to, or on the database partition specified by the DB2NODE environment. For example, to see the contents of the history file use:

```
SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

To hide the syntax of the administrative view, you can create a view as follows:

```
CREATE VIEW LIST_HISTORY AS
SELECT * FROM TABLE(DB_HISTORY()) AS LIST_HISTORY
```

After creating this view, you can run queries against the view. For example:

```
SELECT * FROM LIST_HISTORY
```

or

```
SELECT dbpartitionnum FROM LIST_HISTORY
```

or

```
SELECT dbpartitionnum, start_time, seqnum, tabname, sqlstate
FROM LIST_HISTORY
```

The Table 13 table lists the columns and the column data types returned by the LIST_HISTORY table function.

Table 13. Contents of the history table

Column name	Data type
dbpartitionnum	smallint
EID	bigint
start_time	char(14)
seqnum	smallint
end_time	varchar(14)
firstlog	varchar(254)
lastlog	varchar(254)
backup_id	varchar(24)
tabschema	varchar(128)
tabname	varchar(128)
comment	varchar(254)
cmd_text	clob(2M)
num_tbsps	integer
tbspnames	clob(5M)
operation	char(1)
operationtype	char(1)
objecttype	char(1)

Table 13. Contents of the history table (continued)

Column name	Data type
location	varchar(255)
devicetype	char(1)
entry_status	char(1)
sqlcaid	varchar(8)
sqlcab	integer
sqlcode	integer
sqlerrml	smallint
sqlerrmc	varchar(70)
sqlerrp	varchar(8)
sqlerrd1	integer
sqlerrd2	integer
sqlerrd3	integer
sqlerrd4	integer
sqlerrd5	integer
sqlerrd6	integer
sqlwarn	varchar(11)
sqlstate	varchar(5)

Related reference:

- “DB_HISTORY administrative view – Retrieve history file information” in *Administrative SQL Routines and Views*

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 database instances. You must have a running DAS if you want to use available tools like the Configuration Assistant, the Control Center, or the Development Center.

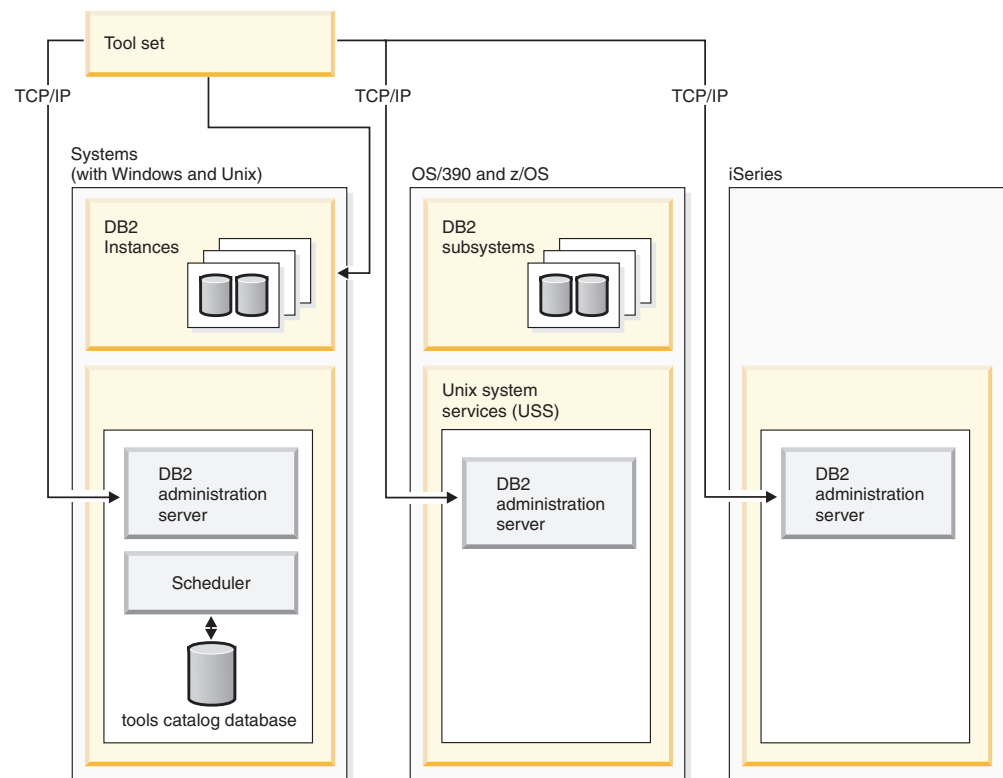


Figure 1. Where DAS is used

DAS assists the Control Center and Configuration Assistant when working on the following administration tasks:

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

the DB2 Discovery utility. This information is used by the Configuration Assistant and the Control Center to simplify and automate the configuration of client connections to DB2 databases.

You can only have one DAS in a database server. If one is already created, you need to drop it by issuing `db2admin drop`. 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 the DB2 system. Products that need DAS, like the Control Center, Replication Center, and Development Center, require the installation of the DAS function. For information on the availability 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 database 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 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:

- “DB2 administration server (DAS) configuration on Enterprise Server Edition (ESE) systems” on page 106
- “DB2 administration server (DAS) first failure data capture (FFDC)” on page 111
- “Discovery of administration servers, instances, and databases” on page 107
- “Security considerations for the DB2 administration server (DAS) on Windows” on page 102

Related tasks:

- “Configuring the DB2 administration server (DAS)” on page 95
- “Creating a DB2 administration server (DAS)” on page 93
- “DB2 administration server (DAS) Java virtual computer setup” on page 101
- “Discovering and hiding server instances and databases” on page 108
- “Listing the DB2 administration server (DAS)” on page 95
- “Notification and contact list setup and configuration” on page 100

- “Removing the DB2 administration server (DAS)” on page 103
- “Setting discovery parameters” on page 109
- “Setting up DB2 administration server (DAS) with Enterprise Server Edition (ESE) systems” on page 104
- “Setting up the DB2 administration server (DAS) to use the Configuration Assistant and the Control Center” on page 110
- “Starting and stopping the DB2 administration server (DAS)” on page 94
- “Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration” on page 96
- “Updating a DB2 administration server (DAS) configuration for discovery” on page 110
- “Updating the DB2 administration server (DAS) on UNIX” on page 102

Creating a DB2 administration server (DAS)

The DB2 administration server (DAS) provides support services for DB2 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**.

Restrictions:

You can only have one DAS in a database server. If one is already created, you need to drop it by issuing `db2admin drop`.

Procedure:

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

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

- On Windows platforms:

Log on to the computer 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 install path:

```
dasCRT -u <DASUser>
```

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

- On AIX:

```
/usr/opt/db2_09_01/instance/  
dasCRT -u <DASUser>
```

- On HP-UX, Solaris operating system, or Linux:

```
/opt/IBM/db2/V9.1/instance/  
dasCRT -u <DASUser>
```

Related tasks:

- “Removing the DB2 administration server (DAS)” on page 103

Related reference:

- “db2admin - DB2 administration server command” in *Command Reference*

Starting and stopping the DB2 administration server (DAS)

Prerequisites:

To manually start or stop the DAS, on Windows you must first log on to the computer 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.

Procedure:

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

When working with the DB2 database manager 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 restart. 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 *Command Reference*
- “dasadm_group - DAS administration authority group name configuration parameter” in *Performance Guide*

Listing the DB2 administration server (DAS)

Procedure:

To obtain the name of the DAS on your computer, 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 *Command Reference*

Configuring the DB2 administration server (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 *Performance Guide*

Related reference:

- “UPDATE ADMIN CONFIGURATION command” in *Command Reference*

Tools catalog database and DB2 administration server (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.

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 computer (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).

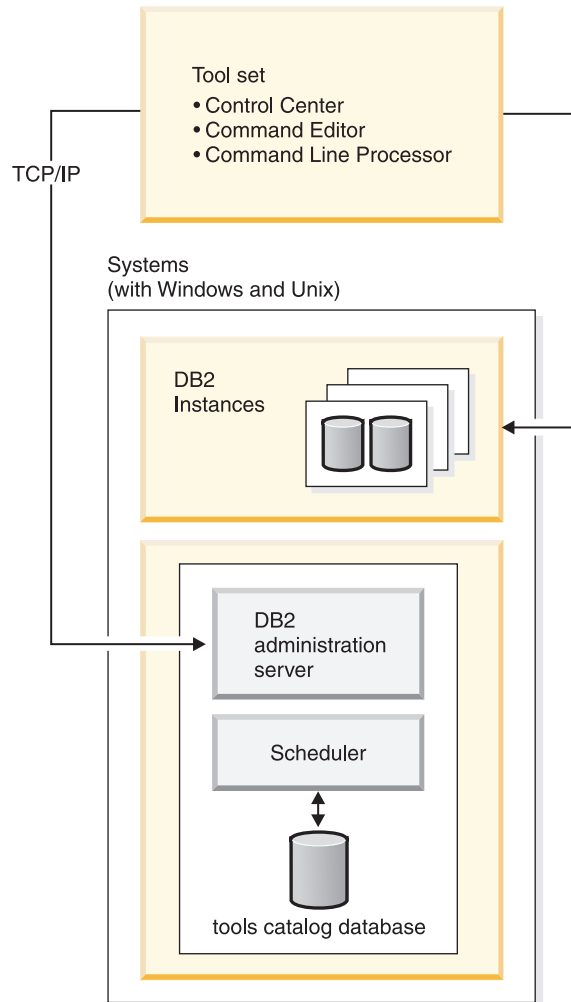


Figure 2. How DAS relates to other parts of DB2 database system

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 want. In this example, the tools catalog database is called “CCMD” and is created under the DB2 database instance on server computer 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 computer 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 might be done during the installation process. If it is done later, it needs to be manually specified to the DAS using a DB2 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 that the DB2 database manager 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 computer C1 (tcp/ip hostname C1).

- 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
```

- 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.

- The tools catalog database can be any name you want. In this example, the tools catalog database is called "CCMD" and is created under the db2inst1 instance on server computer 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".
- 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
```

- 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
```

- 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.
- Assume that the SMTP server used for e-mail notifications is on computer 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 might be done during the installation process. If it is done later, it needs to be manually specified to the DAS using a DB2 Version 8 CLP command as shown above.

- The IBM Software Developer's Kit for Java (SDK) Version 1.3.1 on AIX is installed under /sql1lib/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 /sql1lib/java/jdk
```

- An example AIX client setup.

- Assume that the Control Center is running on client computer C1 (tcp/ip hostname C1).
- 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 ostyle 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:

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

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 using 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 database 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:

- “*contact_host* - Location of contact list configuration parameter” in *Performance Guide*
- “*smtp_server* - SMTP server configuration parameter” in *Performance Guide*

DB2 administration server (DAS) Java virtual computer 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 computer (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 the DB2 database manager 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 Developer’s Kit (SDK) for Java is installed.

Related reference:

- “*jdk_path* - Software Developer's Kit for Java installation path DAS configuration parameter” in *Performance Guide*

- “GET ADMIN CONFIGURATION command” in *Command Reference*
- “UPDATE ADMIN CONFIGURATION command” in *Command Reference*

Security considerations for the DB2 administration server (DAS) on Windows

You might 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 computer using an account or user ID that has local Administrator authority.

Note:

- Recall that passwords are case-sensitive. A mixture of upper and lowercase is allowed which means that the case of the password becomes very important.
- 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 *Command Reference*

Updating the DB2 administration server (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 computer 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 91
- “Security considerations for the DB2 administration server (DAS) on Windows” on page 102

Removing the DB2 administration server (DAS)

Procedure:

To remove the DAS:

- On Windows operating systems:
 1. Log on to the computer 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 database 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 might 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 database manager version and release.

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

Related reference:

- “dasdrop - Remove a DB2 administration server command” in *Command Reference*
- “db2admin - DB2 administration server command” in *Command Reference*

Setting up DB2 administration server (DAS) with Enterprise Server Edition (ESE) systems

The following information shows the steps necessary to configure DB2 Enterprise Server Edition (Linux, Solaris, Windows, HP-UX, and AIX) for remote administration using the Control Center.

During installation, the setup program creates a single DAS on the instance-owning computer. You must create additional DAS on other computers to allow the Control Center or the Configuration Assistant access to other coordinator partitions. The overhead of working as an administrative coordinator partition can then be spread to more than one database partition in an instance. Only if you do not use **db2setup** will you need to do this manually.

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

Procedure:

To distribute the coordinator function:

1. Create a new DAS on the selected additional computers in the partitioned database environment.
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 computer 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 database instance.

Example Environment

product/version:
DB2 UDB ESE V8.1

install path:
install_path

TCP services file:
services

DB2 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: Substitute site-specific values for the fields shown above. For example, the following table contains example pathnames for some sample supported ESE platforms:

Table 14. Example Pathnames for Supported ESE Platforms

Paths	DB2 ESE for AIX	DB2 ESE for Solaris	DB2 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 Enterprise Server Edition, the setup program creates a DAS on the instance-owning computer. The database partition server resides on the same computer as the DAS and is the connection point for the instance. That is, this database partition server is the coordinator partition for requests issued to the instance from the Control Center or the Configuration Assistant.

If DAS is installed on each physical computer, then each computer can act as a coordinator partition. Each physical computer 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 partition functionality and help to balance incoming connections.

Related concepts:

- “DB2 administration server (DAS) configuration on Enterprise Server Edition (ESE) systems” on page 106
- “DB2 Administration Server” on page 91

DB2 administration server (DAS) configuration on Enterprise Server Edition (ESE) systems

The DAS is an administrative control point that performs certain tasks on behalf of the tools. There can be at most one DAS per physical computer. In the case of an ESE instance that consists of several computers, all of the computers 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 database instance (db2inst).

For example, db2inst consists of three nodes distributed across two physical computers or hosts. The minimum requirement can be fulfilled by running **db2as** on hostA and hostB.

Notes:

1. The number of database 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 computer, or physical node, which must be created individually using the **dascrt** command. The DAS on each computer 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 Enterprise Server Edition for Windows, if you are using the Configuration Assistant or the Control Center to automate connection configuration to a DB2 server, the database partition server that is on the same computer 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 Enterprise Server Edition for Windows, creating additional DB2 administration servers on other computers allows the Configuration Assistant or Control Center to configure other systems as coordinator nodes using DB2 Discovery.

When working on DB2 Enterprise Server Edition for Windows, the DB2 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 the DB2 database manager. Therefore, it is not necessary to insert new entries into TCP services file.

Related tasks:

- “Creating a DB2 administration server (DAS)” on page 93

Related reference:

- “db2admin - DB2 administration server command” in *Command Reference*

Discovery of administration servers, instances, and databases

To configure connections to a remote computer, 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 computer, the user would logon to the client computer and run the Configuration Assistant (CA). The CA sends a broadcast signal to all the computers on the network. Any computer 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 computer. 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 computer 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 database 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 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 computer or, if no DNS is configured, a mapping entry in the client's *hosts* file. If you have multiple adapter cards configured on your DB2 server system, you must ensure that TCP/IP is configured on the server to return the correct hostname, and that the DNS or local client's *hosts* file, maps the hostname to the IP address desired.

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

- **KNOWN**

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 181

Related tasks:

- "Discovering and hiding server instances and databases" on page 108
- "Setting discovery parameters" on page 109

Discovering and hiding server instances and databases

You might have multiple instances, and multiple databases within these instances, on a server system. You might want to hide some of these from the Discovery process.

Procedure:

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_db - Discover database configuration parameter” in *Performance Guide*
- “discover_inst - Discover server instance configuration parameter” in *Performance Guide*

Setting discovery parameters

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*.

Procedure:

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** —> **<DB2 copy name>** —> **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:

- “UPDATE ADMIN CONFIGURATION command” in *Command Reference*
- “discover - DAS discovery mode configuration parameter” in *Performance Guide*
- “discover_db - Discover database configuration parameter” in *Performance Guide*
- “discover_inst - Discover server instance configuration parameter” in *Performance Guide*

Setting up the DB2 administration server (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 database partition. When a DAS is created on the database 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 might 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 database partition will respond as a separate DB2SYSTEM name. The actual instances that can be administered depend on the instances known by that physical database partition. Since instances can span multiple database 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 partition.

Related reference:

- “discover - DAS discovery mode configuration parameter” in *Performance Guide*
- “db2ilist - List instances command” in *Command Reference*
- “db2ncrt - Add database partition server to an instance command” in *Command Reference*

Updating a DB2 administration server (DAS) configuration for discovery

Restrictions:

A DB2 administration server (DAS) must reside on each physical database partition in a partitioned database environment. When a DAS is created on the database partition server, 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 partitions when connections are established.

When updating a DAS configuration, and you want to be able to select a coordinating partition from a list of DB2 database 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 environment, the same instance might 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 database systems as coordinator partitions for communications and thereby redistribute the workload.

Related reference:

- "Miscellaneous variables" in *Performance Guide*

DB2 administration server (DAS) first failure data capture (FFDC)

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 product 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 Linux and UNIX 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 91

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 second last 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

You can create a database using the **CREATE DATABASE** command.

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 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. In Version 9.1, the Configuration Advisor is automatically invoked when you create a database. To disable this feature, or to explicitly enable it, use the **db2set** command before creating the database. Examples:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

See Automatic features enabled by default for other DB2 features that are enabled by default.

Prerequisites:

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

The following database privileges are automatically granted to PUBLIC: CREATETAB, BINDADD, CONNECT, IMPLICIT_SCHEMA, and SELECT on the system catalog views. However, if the RESTRICTIVE option is present, no privileges are automatically granted to PUBLIC. For more information on the RESTRICTIVE option, see the documentation on the **CREATE DATABASE** command.

Procedure:

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.

To create a database from a client application, call the `sqlcrea` API.

To create a database using the command line processor, enter: **CREATE DATABASE** <database name>. For example, the following command line processor command creates a database called `person1`, in the default location, with the associated comment "Personnel DB for BSchiefer Co".

```
CREATE DATABASE person1
  WITH "Personnel DB for BSchiefer Co"
```

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.

By default, databases are created in the code page of the application creating them. Therefore, if you create your database from a Unicode (UTF-8) client, your database will be created as a Unicode database. Similarly, if you create your database from an `en_US` (code page 819) client, your database will be created as a single byte US English database.

To override the default code page for the database, it is necessary to specify the desired code set and territory when creating the database. See the **CREATE DATABASE CLP** command or the `sqlcrea` API for information on setting the code set and territory.

In a future release of the DB2 database manager, the default code set will be changed to UTF-8 when creating a database, regardless of the application code page. If a particular code set and territory is needed for a database, then the code set and territory should be specified when the database is created.

Related concepts:

- "Additional database design considerations" in *Administration Guide: Planning*
- "Applications connected to Unicode databases" in *SQL Guide*
- "Automatic features enabled by default" in *Administration Guide: Planning*
- "What to record in a database" in *Administration Guide: Planning*

- “Database authorities” on page 511
- “Multiple instances of the database manager” on page 16

Related tasks:

- “Converting non-Unicode databases to Unicode” in *Administration Guide: Planning*
- “Creating a Unicode database” in *Administration Guide: Planning*
- “Changing node and database configuration files” on page 279
- “Generating recommendations for database configuration” on page 84

Related reference:

- “sqlcrea API - Create database” in *Administrative API Reference*
- “CREATE DATABASE command” in *Command Reference*

Initial database partition groups

When a database is initially created, database partitions are created for all database partitions specified in the db2nodes.cfg file. Other database 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 *Administration Guide: Planning*

Related reference:

- “ADD DBPARTITIONNUM command” in *Command Reference*
- “DROP DBPARTITIONNUM VERIFY command” in *Command Reference*

Creating and managing database partitions and database partition groups

This section describes how to create and manage database partitions and database partition groups.

Creating database partition groups

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 distribution map for the database partition group.
- Generates a distribution map ID.
- Inserts records into the following catalog tables:

- SYSCAT.DBPARTITIONGROUPS
- SYSCAT.PARTITIONMAPS
- SYSCAT.DBPARTITIONGROUPDEF

Prerequisites:

The computers and systems must be available and capable of handling a partitioned database environment. You have purchased and installed DB2 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>)
```

For example, 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 `sqlcrea()` API also create the default system database partition groups, `IBMDEFAULTGROUP`, `IBMCATGROUP`, and `IBMTEMPGROUP`.

Related concepts:

- “Database partition groups” in *Administration Guide: Planning*
- “Distribution maps” in *Administration Guide: Planning*

Related reference:

- “CREATE DATABASE command” in *Command Reference*
- “CREATE DATABASE PARTITION GROUP statement” in *SQL Reference, Volume 2*
- “sqlcrea API - Create database” in *Administrative API Reference*

Managing database partitions

 You can use the Partitions view in the Control Center to perform the following tasks:

- Start partitions
- Stop partitions
- Drop partitions

- Trace partitions
- Display the diagnostics log

If requested to do so from IBM Support, run the trace utility using the options that they recommend. The trace utility records information about DB2 operations and formats this information into readable form. For more information, see `db2trc - Trace: DB2` topic.

Attention: Only use the trace facility when directed by DB2 Customer Service or by a technical support representative to do so.

Use the Diagnostic Log window to view text information logged by the DB2 trace utility.

The Partitions view displays the following information:

Node Number

This column contains icons and node numbers. The node numbers are unique numbers, and can be from 0 to 999. The numbers are stored in the `db2nodes.cfg` file. Node numbers are displayed in ascending sequence, though there might be gaps in the sequence.

Node numbers, once assigned, cannot be changed. This safeguard ensures that the information in the distribution map (which details how data is partitioned) is not compromised.

Host Name

The host name is the IP address used by fast communication manager (FCM) for internal communications. (However, if a switch name is specified, FCM uses the switch name. In this situation, the host name is used only for `DB2START`, `DB2STOP`, and `db2_all`.) The host name is stored in the `db2nodes.cfg` file.

Port Number

The port number is 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. This number is stored in the `db2nodes.cfg` file.

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

For each displayed host name, one port number will be 0. Port number 0 indicates the default node on the host to which clients connect. (To override this behavior, use the `DB2NODE` environment variable in `db2profile` script.)

Switch Name

The switch name is used to support a host that has more than one active TCP/IP interface, each with its own host name. The switch name is stored in the `db2nodes.cfg` file.

The switch name is only used for RS/6000 SP machines that have a primary host name that is either an Ethernet or a token-ring name, and DB2 Universal Database Enterprise Server Edition is using the alternative switch name. If the switch name was not specified in the `db2nodes.cfg` file, it is the same as the host name.

Prerequisites:

To work with database partitions, you will need authority to attach to an instance. Anyone with SYSADM or DBADM authority can grant you with the authority to access a specific instance.

To view the DB2 logs, you will need authority to attach to an instance. Anyone with SYSADM or DBADM authority can grant you with the authority to access a specific instance.

Procedure:

- Open the Partitions view: From the Control Center, expand the object tree until you find the instance for which you want to view the partitions. Right-click on the instance you want and select **Open→Partitions** from the pop-up menu. The Partitions view opens.
- To start partitions: Highlight one or more partitions and select **Partitions→Start**. The selected partitions are started.
- To stop partitions: Highlight one or more partitions and select **Partitions→Stop**. The selected partitions are stopped.
- To run the trace utility on a partition:
 1. Open the DB2 Trace window: Highlight a partition, and select **Partitions→Service→Trace**. The DB2 Trace window opens.
 2. Specify the trace options.
 3. Click **Start** to start recording information and **Stop** to stop recording information.
 4. Optional: View the trace output and the DB2 logs.
 5. Send the trace output to IBM Support, if requested to do so.

Related concepts:

- “Adding database partitions in a partitioned database environment” on page 123
- “Attributes of detached data partitions” on page 354
- “Partitioned database environments” in *Administration Guide: Planning*
- “Partitioned database authentication considerations” on page 496
- “Partitioned databases” in *Administration Guide: Planning*

Related tasks:

- “Adding a database partition to a running database system” on page 119
- “Adding a database partition to a stopped database system on Windows” on page 122
- “Adding a database partition to a stopped database system on UNIX” on page 120
- “Adding database partitions to an instance using the Add Partitions wizard” on page 124
- “Adding data partitions to partitioned tables” on page 356
- “Adding database partitions using the Add Partitions launchpad” on page 125
- “Attaching a data partition” on page 346
- “Changing the database configuration across multiple database partitions” on page 281
- “Creating a table in a partitioned database environment” on page 191

Adding and dropping database partitions

This section describes how to add or drop database partitions on UNIX and Windows platforms.

Adding a database partition to a running database system

You can add new database partitions to a partitioned database environment while it is running and while applications are connected to databases. However, a newly added server does not become available to all databases until the database manager is shut down and restarted.

Procedure:

To add a database partition to a running database manager using the Control Center:

1. Open the Add Partitions wizard:
 - a. From the Control Center, expand the object tree until you find the instance object that you want to work with. Right-click the object, and click **Add Partitions** from the pop-up menu. The Add Partitions launchpad opens.
 - b. Click the **Add Partitions** button. The Add Partitions wizard opens.
2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is available when you complete enough information for the wizard to add the partition.

To add a database partition to a running database manager using the command line:

1. On any existing database partition, run the DB2START command.

On all platforms, specify the new database partition values for DBPARTITIONNUM, ADD DBPARTITIONNUM, HOSTNAME, PORT, and NETNAME parameters. On the Windows platform, you also specify the COMPUTER, USER, and PASSWORD parameters.

You can also specify the source for any temporary table space container definitions that need to be created with the databases. If you do not provide table space information, temporary table space container definitions are retrieved from the catalog partition for each database.

When the DB2START command is complete, the new server is stopped.
2. Stop the database manager on all database partitions by running the DB2STOP command.

When you stop all the database partitions in the system, the node configuration file is updated to include the new database partition. The node configuration file is not updated with the new server information until DB2STOP is executed. This ensures that the ADD DBPARTITIONNUM command, which is called when you specify the ADDNODE parameter to the DB2START command, runs on the correct database partition. When the utility ends, the new server partition is stopped.
3. Start the database manager by running the DB2START command.

The newly added database partition is now started along with the rest of the system.

When all the database partitions in the system are running, you can run system-wide activities, such as creating or dropping a database.

Note: You might have to issue the DB2START command twice for all database partition servers to access the new db2nodes.cfg file.

4. Back up all databases on the new database partition. (Optional)
5. Redistribute data to the new database partition. (Optional)

Related concepts:

- “Adding database partitions in a partitioned database environment” on page 123

Related tasks:

- “Adding a database partition to a stopped database system on UNIX” on page 120
- “Adding a database partition to a stopped database system on Windows” on page 122

Adding a database partition to a stopped database system on UNIX

You can add new database partitions to a partitioned database system while it is stopped. The newly added database partition becomes available to all databases when the database manager is started up again.

Prerequisites:

You must install the new server if it does not exist, including the following tasks:

- Making executables accessible (using shared file-system mounts or local copies)
- Synchronizing operating system files with those on existing processors
- Ensuring that the sql1ib directory is accessible as a shared file system
- Ensuring that the relevant operating system parameters (such as the maximum number of processes) are set to the appropriate values

You must also register the host name with the name server or in the hosts file in the etc directory on all database partitions.

Procedure:

To add a database partition to a stopped partitioned database server using the Control Center:

1. Open the Add Partitions wizard:
 - a. From the Control Center, expand the object tree until you find the instance object that you want to work with. Right-click the object, and click **Add Partitions** from the pop-up menu. The Add Partitions launchpad opens.
 - b. Click the **Add Partitions** button. The Add Partitions wizard opens.
2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is available when you complete enough information for the wizard to add the partition.

To add a database partition to a stopped partitioned database server using the command line:

1. Issue DB2STOP to stop all the database partitions.
2. Run the ADD DBPARTITIONNUM command on the new server.

A database partition is created locally for every database that already exists in the system. The database parameters for the new database partitions are set to the default value, and each database partition remains empty until you move data to it. Update the database configuration parameter values to match those on the other database partitions.

3. Run the DB2START command to start the database system. Note that the node configuration file (cfg) has already been updated to include the new server during the installation of the new server.
4. Update the configuration file on the new database partition as follows:
 - a. On any existing database partition, run the DB2START command.
Specify the new database partition values for DBPARTITIONNUM, ADDDBPARTITIONNUM, HOSTNAME, PORT, and NETNAME parameters as well as the COMPUTER, USER, and PASSWORD parameters.
You can also specify the source for any temporary table space container definitions that need to be created with the databases. If you do not provide table space information, temporary table space container definitions are retrieved from the catalog partition for each database.
When the DB2START command is complete, the new server is stopped.
 - b. Stop the entire database manager by running the DB2STOP command.
When you stop all the database partitions in the system, the node configuration file is updated to include the new database partition. The node configuration file is not updated with the new server information until DB2STOP is executed. This ensures that the ADD DBPARTITIONNUM command, which is called when you specify the ADDDBPARTITIONNUM parameter to the DB2START command, runs on the correct database partition. When the utility ends, the new server partition is stopped.
5. Start the database manager by running the DB2START command.
The newly added database partition is now started with the rest of the system.
When all the database partitions in the system are running, you can run system-wide activities, such as creating or dropping a database.

Note: You might have to issue the DB2START command twice for all database partition servers to access the new db2nodes.cfg file.

6. Back up all databases on the new database partition. (Optional)
7. Redistribute data to the new database partition. (Optional)

You can also update the configuration file manually, as follows:

1. Edit the db2nodes.cfg file and add the new database partition to it.
2. Issue the following command to start the new database partition: DB2START DBPARTITIONNUM partitionnum
Specify the number you are assigning to the new database partition as the value of nodenum.
3. If the new server is to be a logical partition (that is, it is not database partition 0), use db2set command to update the DBPARTITIONNUM registry variable. Specify the number of the database partition you are adding.
4. Run the ADD NODE command on the new database partition.
This command creates a database partition locally for every database that already exists in the system. The database parameters for the new database partitions are set to the default value, and each database partition remains empty until you move data to it. Update the database configuration parameter values to match those on the other database partitions.

5. When the ADD DBPARTITIONNUM command completes, issue the DB2START command to start the other database partitions in the system.

Do not perform any system-wide activities, such as creating or dropping a database, until all database partitions are successfully started.

Related concepts:

- “Error recovery when adding database partitions” on page 128

Related tasks:

- “Adding a database partition to a running database system” on page 119
- “Adding a database partition to a stopped database system on Windows” on page 122
- “Dropping a database partition” on page 125

Adding a database partition to a stopped database system on Windows

You can add new database partitions to a partitioned database system while it is stopped. The newly added database partition becomes available to all databases when the database manager is started up again.

Prerequisites:

You must install the new server before you can create a database partition on it.

Procedure:

To add a database partition to a stopped partitioned database server using the Control Center:

1. Open the Add Partitions wizard:
 - a. From the Control Center, expand the object tree until you find the instance object that you want to work with. Right-click the object, and click **Add Partitions** from the pop-up menu. The Add Partitions launchpad opens.
 - b. Click the **Add Partitions** button. The Add Partitions wizard opens.
2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is available when you complete enough information for the wizard to add the partition.

To add a database partition to a stopped partitioned database server using the command line:

1. Issue DB2STOP to stop all the database partitions.
2. Run the ADD DBPARTITIONNUM command on the new server.

A database partition is created locally for every database that already exists in the system. The database parameters for the new database partitions are set to the default value, and each database partition remains empty until you move data to it. Update the database configuration parameter values to match those on the other database partitions.
3. Run the DB2START command to start the database system. Note that the node configuration file (cfg) has already been updated to include the new server during the installation of the new server.
4. Update the configuration file on the new database partition as follows:

- a. On any existing database partitions, run the DB2START command.
Specify the new database partition values for DBPARTITIONNUM, ADDDBPARTITIONNUM, HOSTNAME, PORT, and NETNAME parameters as well as the COMPUTER, USER, and PASSWORD parameters.
You can also specify the source for any temporary table space container definitions that need to be created with the databases. If you do not provide table space information, temporary table space container definitions are retrieved from the catalog partition for each database.
When the DB2START command is complete, the new server is stopped.
 - b. Stop the entire database manager by running the DB2STOP command.
When you stop all the database partitions in the system, the node configuration file is updated to include the new database partition. The node configuration file is not updated with the new server information until DB2STOP is executed. This ensures that the ADD DBPARTITIONNUM command, which is called when you specify the ADDDBPARTITIONNUM parameter to the DB2START command, runs on the correct database partition. When the utility ends, the new server partition is stopped.
5. Start the database manager by running the DB2START command.
The newly added database partition is now started with the rest of the system.
When all the database partitions in the system are running, you can run system-wide activities, such as creating or dropping a database.

Note: You might have to issue the DB2START command twice for all database partition servers to access the new db2nodes.cfg file.
 6. Back up all databases on the new database partition. (Optional)
 7. Redistribute data to the new database partition. (Optional)

Related concepts:

- “Error recovery when adding database partitions” on page 128
- “Adding database partitions in a partitioned database environment” on page 123

Related tasks:

- “Adding a database partition to a running database system” on page 119
- “Adding a database partition to a stopped database system on UNIX” on page 120

Adding database partitions in a partitioned database environment

You can add database partitions to the partitioned database system either when it is running, or when it is stopped. Because adding a new server can be time consuming, you may want to do it when the database manager is already running.

Use the ADD DBPARTITIONNUM command to add a database partition to a system. This command can be invoked in the following ways:

- As an option on db2start
- With the command-line processor ADD DBPARTITIONNUM command
- With the API function sqladdn
- With the API function sqlstart

If your system is stopped, you use db2start. If it is running, you can use any of the other choices.

When you use the ADD DBPARTITIONNUM command to add a new database partition to the system, all existing databases in the instance are expanded to the new database partition. You can also specify which containers to use for temporary table spaces for the databases. The containers can be:

- The same as those defined for the catalog partition for each database. (This is the default.)
- The same as those defined for another database partition.
- Not created at all. You must use the ALTER TABLESPACE statement to add temporary table space containers to each database before the database can be used.

You cannot use a database on the new database partition to contain data until one or more database partition groups are altered to include the new database partition.

You cannot change from a single-partition database to a multi-partition database by simply adding a database partition to your system. This is because the redistribution of data across database partitions requires a distribution key on each affected table. The distribution keys are automatically generated when a table is created in a multi-partition database. In a single-partition database, distribution keys can be explicitly created with the CREATE TABLE or ALTER TABLE SQL statements.

Note: If no databases are defined in the system and you are running Enterprise Server Edition on a UNIX operating system, edit the db2nodes.cfg file to add a new database partition definition; do not use any of the procedures described, as they apply only when a database exists.

Windows Considerations: If you are using Enterprise Server Edition on Windows and have no databases in the instance, use the DB2NCRT command to scale the database system. If, however, you already have databases, use the DB2START ADDNODE command to ensure that a database partition is created for each existing database when you scale the system. On Windows, you should never manually edit the node configuration file (db2nodes.cfg), as this can introduce inconsistencies into the file.

Related tasks:

- “Adding a database partition to a running database system” on page 119
- “Adding a database partition to a stopped database system on Windows” on page 122
- “Dropping a database partition” on page 125

Adding database partitions to an instance using the Add Partitions wizard

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

Prerequisites:

To work with database partition groups, you must have SYSADM or DBADM authority.

Procedure:

To add partitions to an instance:

1. Open the Add Partitions wizard:
 - a. From the Control Center, expand the object tree until you find the instance object that you want to work with. Right-click the object, and click Add Partitions in the pop-up menu. The Add Partitions launchpad opens.
 - b. Click the **Add Partitions** button. The Add Partitions wizard opens.
2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is available when you complete enough information for the wizard to add the partition.

Related concepts:

- “Partitioned databases” in *Administration Guide: Planning*

Adding database partitions using the Add Partitions launchpad

Use the Add Partitions launchpad to guide you through the tasks necessary to add partitions to an instance. The launchpad also helps you assign the newly added partitions to database partition groups in the databases that are in the instance, and then redistribute data in the database partition groups.

It is recommended that you backup all databases in the instance before and after redistributing data in database partition groups. If you do not back up your databases, you might corrupt the databases and you might not be able to recover them.

Procedure:

To add partitions:

1. Optional: Back up the database.
2. Open the Add Partitions launchpad: From the Control Center, expand the object tree until you find the instance object that you want to work with. Right-click the object, and click **Add Partitions** in the pop-up menu. The Add Partitions launchpad opens.
3. Add partitions.
4. Redistribute data.
5. Optional: Back up the database.

Related tasks:

- “Backing up data using the Backup wizard” on page 387
- “Redistributing data in a database partition group” on page 128
- “Adding database partitions to an instance using the Add Partitions wizard” on page 124

Dropping a database partition

You can drop a database partition that is not being used by any database and free the computer for other uses.

Prerequisites:

Verify that the database partition is not in use by issuing the DROP DBPARTITION VERIFY command or the *sqledrpn* API.

- If you receive message SQL6034W (Database partition not used in any database), you can drop the database partition.
- If you receive message SQL6035W (Database partition in use by database), use the **REDISTRIBUTE NODEGROUP** command to redistribute the data from the database partition that you are dropping to other database partitions from the database alias.

Also ensure that all transactions for which this database partition was the coordinator have all committed or rolled back successfully. This might require doing crash recovery on other servers. For example, if you drop the coordinator partition, and another database partition participating in a transaction crashed before the coordinator partition was dropped, the crashed database partition will not be able to query the coordinator partition for the outcome of any in-doubt transactions.

Procedure:

To drop a database partition using the Control Center:

1. Optional: Back up the database.
2. Open the Drop Database Partitions launchpad. To open the Drop Database Partitions launchpad:
 - a. Open the Database Partitions view. To open the Database Partitions view: From the Control Center, expand the object tree until you find the instance for which you want to view the database partitions. Right-click on the instance you want and select **Open->Database Partitions** from the pop-up menu. The Database Partitions view opens for the selected instance.
 - b. Select the database partitions you want to drop.
 - c. Right-click the selected database partitions and click **Drop** in the pop-up menu. The Drop Database Partitions launchpad opens.
3. Drop the database partitions from database partition groups.
Note: This operation does not drop the database partitions immediately. Instead, it flags the database partitions that you want to drop so that data can be move off them when you redistribute the data in the database partition group.

To drop a database partition using the command line processor:

1. Issue the **DB2STOP** command with the DROP NODENUM parameter to drop the database partition. After the command completes successfully, the system is stopped.
2. Start the database manager with the **DB2START** command.

Related concepts:

- “Management of database server capacity” on page 29
- “Adding database partitions in a partitioned database environment” on page 123

Related reference:

- “DROP DBPARTITIONNUM VERIFY command” in *Command Reference*
- “sqledrpn API - Check whether a database partition server can be dropped” in *Administrative API Reference*

Dropping database partitions from the instance using the Drop Partitions launchpad

Use the Drop Partitions launchpad to guide you through the tasks necessary to drop database partitions from database partition groups, redistribute data in database partition groups, and drop partitions from an instance.

Note: When you drop database partitions from database partition groups the database partitions are not immediately dropped. Instead, the database partitions that you want to drop are flagged so that data can be move off them when you redistribute the data in the database partition groups.

It is recommended that you backup all databases in the instance before and after redistributing data in database partition groups. If you do not back up your databases, you might corrupt the databases and you might not be able to recover them.

Procedure:

To drop partitions using the Drop Partitions launchpad:

1. Optional: Back up the database.
2. Open the Drop Partitions launchpad:
 - a. Open the Partitions window: From the Control Center, expand the object tree until you find the instance for which you want to view the partitions. Right-click on the instance you want and select **Open->Partitions** from the pop-up menu. The Partitions window opens for the selected instance.
 - b. Select the partitions you want to drop.
 - c. Right-click the selected partitions and click **Drop** in the pop-up menu. The Drop Partitions launchpad opens.
3. Drop the database partitions from database partition groups:
 - a. Confirm the database partitions you want to drop from the database partition groups.

Note:

- You must drop the database partitions from database partition groups before you drop partitions from the instance.
 - This operation does not drop the database partitions immediately. Instead, it flags the database partitions that you want to drop so that data can be move off them when you redistribute the data in the database partition group.
4. Redistribute data.
 5. Drop partitions from the instance:
 - a. Open the Drop Partitions from Instance Confirmation window:
 - Open the Partitions window as described above.
 - Select the partitions you want to drop.
 - Right-click the selected partitions and click Drop in the pop-up menu. The Drop Partitions launchpad opens.
 - Click the **Drop Partitions from Instance** push button. The Drop Partitions from Instance Confirmation window opens.
 - b. In the **Drop** column, verify that you want to drop the partitions for the selected instance.

- c. Click **OK** to open a window where you can schedule when you want to drop the partition.
6. Optional: Back up the database.

Related concepts:

- “Partitioned databases” in *Administration Guide: Planning*

Related tasks:

- “Backing up data using the Backup wizard” on page 387
- “Redistributing data in a database partition group” on page 128

Redistributing data in a database partition group

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

Prerequisites:

To work with database partition groups, you must have SYSADM or DBADM authority.

Procedure:

To redistribute data in your database partition group:

1. Open the Redistribute Data wizard: From the Control Center, expand the object tree until you find the **Database Partition Groups** folder. Any existing database partition groups are displayed in the contents pane on the right side of the window. Right-click on the database partition group that you want to work with and select **Redistribute** from the pop-up menu. The Redistribute Data wizard opens.

You can also open the Redistribute Data wizard from the Add Partitions launchpad or the Drop Partitions launchpad.

2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is enabled when you specify enough information for the wizard to redistribute your data.

Related concepts:

- “Logs” in *Administration Guide: Planning*
- “Partitioned databases” in *Administration Guide: Planning*

Related tasks:

- “Adding database partitions using the Add Partitions launchpad” on page 125
- “Dropping database partitions from the instance using the Drop Partitions launchpad” on page 127

Error recovery when adding database partitions

In version 8.1 and later, adding database partitions does not fail as a result of non-existent buffer pools because DB2 creates system buffer pools to provide default automatic support for all buffer-pool page sizes. However, if one of these system buffer pools is used, performance might be seriously affected because the

system buffer pools are very small. If a system buffer pool is used, a message is written to the administration notification log.

System buffer pools are used in database partition addition scenarios in the following circumstances:

- You add database partitions to a partitioned database environment that has one or more system temporary table spaces with a page size that is different from the default of 4 KB. When a database partition is created, only the IBMDEFAULTDP buffer pool exists, and this buffer pool has a page size of 4 KB.

Consider the following examples:

1. You use the `db2start` command to add a database partition to the current multi-partition database:

```
DB2START DBPARTITIONNUM 2 ADD DBPARTITIONNUM HOSTNAME newhost PORT 2
```

2. You use the `ADD DBPARTITIONNUM` command after you manually update the `db2nodes.cfg` file with the new database partition description.

One way to prevent these problems is to specify the `WITHOUT TABLESPACES` clause on the `ADD NODE` or the `db2start` command. After doing this, you need to use the `CREATE BUFFERPOOL` statement to create the buffer pools using `,` and associate the system temporary table spaces to the buffer pool using the `ALTER TABLESPACE` statement.

- You add database partitions to an existing database partition group that has one or more table spaces with a page size that is different from the default page size, which is 4 KB. This occurs because the non-default page-size buffer pools created on the new database partition have not been activated for the table spaces.

Note: In previous versions of DB2, this command used the `NODEGROUP` keyword instead of the `DATABASE PARTITION GROUP` keywords.

Consider the following example:

- You use the `ALTER DATABASE PARTITION GROUP` statement to add a database partition to a database partition group, as follows:

```
DB2START
CONNECT TO mpp1
ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
```

One way to prevent this problem is to create buffer pools for each page size and then to reconnect to the database before issuing the following `ALTER DATABASE PARTITION GROUP` statement:

```
DB2START
CONNECT TO mpp1
CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
CONNECT RESET
CONNECT TO mpp1
ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
```

Note: If the database partition group has table spaces with the default page size, message `SQL1759W` is returned:

Related tasks:

- “Adding a database partition to a running database system” on page 119
- “Adding a database partition to a stopped database system on Windows” on page 122

Issuing commands to multiple database partitions

This section describes how to issue commands to multiple database partitions, including problem resolution.

Issuing commands in a partitioned database environment

In a partitioned database environment, you might want to issue commands to be run on computers 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 computers 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 environments only.

Notes:

1. On Linux and UNIX 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. Also, on Linux and UNIX platforms, **rah** uses the remote shell program specified by the `DB2RSHCMD` registry variable. You can select between the two remote shell programs: `ssh` (for additional security), or `rsh` (or `remsh` for HP-UX). The `ssh` remote shell program is used to prevent the transmission of passwords in clear text in UNIX operating system environments. If this registry variable is not set, `rsh` (or `remsh` for HP-UX) is used.
3. On Windows, 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*, which 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 computer. If you want to run **db2trc** on all logical nodes on all computers, use **rah**.

Related concepts:

- “rah and db2_all commands overview” on page 130
- “Specifying the rah and db2_all commands” on page 132

Related reference:

- “rah and db2_all command descriptions” on page 131

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 Linux and UNIX 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 computer where the command is issued. On Windows, if you run the commands in parallel, the output is displayed at the computer 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 Linux and UNIX platforms, you separate multiple commands using a semicolon (;). On Windows, 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 ";DB2 UPDATE DB CFG FOR sample USING LOGFILSIZ 100"
```

Related concepts:

- “Issuing commands in a partitioned database environment” on page 130
- “Specifying the rah and db2_all commands” on page 132

Related reference:

- “rah and db2_all command descriptions” on page 131

rah and db2_all command descriptions

You can use the following commands:

Command	Description
rah	Runs the command on all computers.
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 <i>not</i> issue this command except under direction from IBM service.
db2_call_stack	<p>On Linux and UNIX platforms, causes all processes running on all database partition servers to write call traceback to the syslog.</p> <p>On Windows, causes all processes running on all database partition servers to write call traceback to the <code>Pxxxx.mnn</code> file in the instance directory, where <code>Pxxxx</code> is the process ID and <code>mnn</code> is the database partition number.</p>

On Linux and UNIX platforms, these commands execute **rah** with certain implicit settings such as:

- Run in parallel at all computers
- Buffer command output in `/tmp/$USER/db2_kill`, `/tmp/$USER/db2_call_stack` respectively.

On Windows, these commands execute **rah** to run in parallel at all computers.

Related concepts:

- “rah and db2_all commands overview” on page 130
- “Running commands in parallel on Linux and UNIX platforms” on page 133
- “Specifying the rah and db2_all commands” on page 132

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 Linux and UNIX 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 Linux and UNIX 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 Linux and UNIX 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, 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:

- “Additional rah information (Solaris and AIX only)” on page 135
- “Running commands in parallel on Linux and UNIX platforms” on page 133

Related tasks:

- “Setting the default environment profile for rah on Windows” on page 141

Related reference:

- “Controlling the rah command” on page 139
- “rah and db2_all command descriptions” on page 131
- “rah command prefix sequences” on page 135

Running commands in parallel on Linux and UNIX platforms

Note: The information in this section applies to Linux and UNIX platforms only.

By default, the command is run sequentially at each computer, 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 computer. This process retrieves the output in two pieces:

1. After the remote command completes.
2. After the rshell terminates, which might 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 computers:

- Ensure that a directory /tmp/\$USER exists for your user ID at each computer. 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 computer 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 computer ID at which you run the remote command.

Related concepts:

- “Additional rah information (Solaris and AIX only)” on page 135

Related tasks:

- “Monitoring rah processes on Linux and UNIX platforms” on page 134

Related reference:

- “Determining problems with rah on Linux and UNIX platforms” on page 141
- “rah command prefix sequences” on page 135

Monitoring rah processes on Linux and UNIX platforms

Procedure:

Note: The information in this section applies to Linux and UNIX 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 might 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 Linux and UNIX 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:

- “Additional rah information (Solaris and AIX only)” on page 135
- “Running commands in parallel on Linux and UNIX platforms” on page 133

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 Linux and UNIX platforms” on page 133

Related tasks:

- “Monitoring rah processes on Linux and UNIX platforms” on page 134

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 Linux and UNIX platforms:
`rah "};ps -F pid,ppid,etime,args -u $USER"`
- On Windows:
`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 might be later if, for example, child processes (on Linux and UNIX 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 computer.

Note: On Linux and UNIX 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 Linux and UNIX platforms, specifying & degrades performance, because more **rsh** commands are required.
- ;; Same as ||& above. This is an alternative shorter form.

Note: On Linux and UNIX 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 Linux and UNIX platforms only.
- } Prepends dot-execution of file named in \$RAHENV (probably .kshrc) before executing command.

Note: Available on Linux and UNIX 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 Linux and UNIX platforms only.
-) Suppresses execution of user's profile and of file named in \$RAHENV.

Note: Available on Linux and UNIX platforms only.
- ' Echoes the command invocation to the computer.
- < Sends to all the computers 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 database partition 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 computer name.

" Substitutes occurrences of () by the computer index, and substitutes occurrences of ## by the database partition number.

Notes:

1. The computer index is a number that associated with a computer in the database system. If you are not running multiple logical partitions, the computer index for a computer corresponds to the database partition number for that computer in the node configuration file. To obtain the computer index for a computer in a multiple logical partition database environment, do not count duplicate entries for those computers that run multiple logical partitions. For example, if MACH1 is running two logical partitions and MACH2 is also running two logical partitions, the database partition number for MACH3 is 5 in the node configuration file. The computer index for MACH3, however, would be 3.

On Windows, do not edit the node configuration file. To obtain the computer index, use the **db2nlist** command.

2. When " is specified, duplicates are not eliminated from the list of computers.

When using the <<-nnn< and <<+nnn< prefix sequences, *nnn* is any 1-, 2- or 3-digit database 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:

- "Running commands in parallel on Linux and UNIX platforms" on page 133
- "Specifying the rah and db2_all commands" on page 132

Related reference:

- "rah and db2_all command descriptions" on page 131

Specifying the list of computers in a partitioned database environment

Procedure:

By default, the list of computers 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 computers by exporting (on Linux and UNIX platforms) or setting (on Windows) the environment variable `RAHOSTFILE`.
- Specifying the list explicitly, as a string of names separated by spaces, by exporting (on Linux and UNIX platforms) or setting (on Windows) the environment variable `RAHOSTLIST`.

Note: If both of these environment variables are specified, `RAHOSTLIST` takes precedence.

Note: On Windows, to avoid introducing inconsistencies into the node configuration file, do *not* edit it manually. To obtain the list of computers in the instance, use the `db2nlist` command.

Related tasks:

- “Eliminating duplicate entries from a list of computers in a partitioned database environment” on page 138

Eliminating duplicate entries from a list of computers in a partitioned database environment

Procedure:

If you are running DB2 Enterprise Server Edition with multiple logical database partition servers on one computer, your `db2nodes.cfg` file will contain multiple entries for that computer. In this situation, the `rah` command needs to know whether you want the command to be executed once only on each computer or once for each logical database partition listed in the `db2nodes.cfg` file. Use the `rah` command to specify computers. Use the `db2_all` command to specify logical database partitions.

Note: On Linux and UNIX platforms, if you specify computers, `rah` will normally eliminate duplicates from the computer list, with the following exception: if you specify logical database partitions, `db2_all` prepends the following assignment to your command:

```
export DB2NODE=nnn (for Korn shell syntax)
```

where `nnn` is the database partition 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 database partitions, you can restrict the list to include all logical database partitions except one, or only specify one database partition server using the `<<-nnn<` and `<<+nnn<` prefix sequences. You might want to do this if you want to run a command to catalog the database partition 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 database partition number of the catalog partition to do this.

If you execute `db2 restart database` using the `rah` command, duplicate entries are eliminated from the list of computers. 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 computer.

Related tasks:

- "Specifying the list of computers in a partitioned database environment" on page 137

Related reference:

- "RESTART DATABASE command" in *Command Reference*
- "rah command prefix sequences" on page 135

Controlling the rah command

You can use the following environment variables to control the **rah** command.

Table 15.

Name	Meaning	Default
\$RAHBUFDIR Note: Available on Linux and UNIX platforms only.	Directory for buffer	/tmp/\$USER
\$RAHBUFNAME Note: Available on Linux and UNIX platforms only.	Filename for buffer	rahout
\$RAHOSTFILE (on Linux and UNIX platforms); RAHOSTFILE (on Windows)	File containing list of hosts	db2nodes.cfg
\$RAHOSTLIST (on Linux and UNIX platforms); RAHOSTLIST (on Windows)	List of hosts as a string	extracted from \$RAHOSTFILE
\$RAHCHECKBUF Note: Available on Linux and UNIX platforms only.	If set to "no", bypass checks	not set
\$RAHSLEEPTIME (on Linux and UNIX platforms); RAHSLEEPTIME (on Windows)	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

Table 15. (continued)

Name	Meaning	Default
\$RAHWAITTIME (on Linux and UNIX platforms); RAHWAITTIME (on Windows)	<p>On Windows, interval in seconds between successive checks that remote jobs are still running.</p> <p>On Linux and UNIX platforms, interval in seconds between successive checks that remote jobs are still running and rah: waiting for <pid> ... messages.</p> <p>On all platforms, specify any positive integer. Prefix value with a leading zero to suppress messages, for example, export RAHWAITTIME=045.</p> <p>It is not necessary to specify a low value as rah does not rely on these checks to detect job completion.</p>	45 seconds
\$RAHENV Note: Available on Linux and UNIX platforms only.	Specifies filename to be executed if \$RAHDOTFILES=E or K or PE or B	\$ENV
\$RAHUSER (on Linux and UNIX platforms); RAHUSER (on Windows)	<p>On Linux and UNIX platforms, user ID under which the remote command is to be run.</p> <p>On Windows, the logon account associated with the DB2 Remote Command Service</p>	\$USER

Note: On Linux and UNIX 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 Linux and UNIX platforms” on page 140

Using \$RAHDOTFILES on Linux and UNIX platforms

Note: The information in this section applies to Linux and UNIX 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 139

Setting the default environment profile for **rah** on Windows

Procedure:

Note: The information in this section applies to Windows® 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 132

Determining problems with **rah** on Linux and UNIX platforms

Note: The information in this section applies to Linux and UNIX platforms only. Here are suggestions on how to handle some problems that you might encounter when you are running **rah**:

1. **rah** hangs (or takes a very long time)

This problem might 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 computers to check the existence of the buffer directory, and to create it if it does not exist.
 - One or more of the computers 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 computers does not have the ID running **rah** correctly defined in its `.hosts` file, or the ID running **rah** does not have one of the computers correctly defined in its `.rhosts` file. If the `DB2RSHCMD` registry variable has been configured to use `ssh`, then the `ssh` clients and servers on each computer might not be configured correctly.

Note: You might have a need to have greater security regarding the transmission of passwords in clear text between database partitions. This will depend on the remote shell program you are using. **rah** uses the remote shell program specified by the `DB2RSHCMD` registry variable. You can select between the two remote shell programs: `ssh` (for additional security), or `rsh` (or `remsh` for HP-UX). If this registry variable is not set, `rsh` (or `remsh` for HP-UX) is used.

3. When running commands in parallel using background remote shells, although the commands run and complete within the expected elapsed time at the computers, **rah** takes a long time to detect this and put up the shell prompt. The ID running **rah** does not have one of the computers correctly defined in its `.rhosts` file, or if the `DB2RSHCMD` registry variable has been configured to use `ssh`, then the `ssh` clients and servers on each computer might not be configured correctly.
4. Although **rah** runs fine when run from the shell command line, if you run **rah** remotely using `rsh`, for example,

```
rsh somewhere -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 /sql11ib/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 remote shell commands and therefore degrades performance.

Related reference:

- “Controlling the rah command” on page 139

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)” on page 144
- “Changing the database partition (Windows)” on page 145
- “Dropping a database partition from an instance (Windows)” on page 147

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 database partition server by using:

```
db2nlist /s
```

The status of each database partition server might be one of: starting, running, stopping, or stopped.

Related tasks:

- “Adding a database partition server to an instance (Windows)” on page 144

- “Changing the database partition (Windows)” on page 145
- “Dropping a database partition from an instance (Windows)” on page 147

Adding a database partition server to an instance (Windows)

Procedure:

On Windows, use the **db2ncrt** command to add a database partition server 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 might cause inconsistencies in the partitioned database environment.

The command has the following required parameters:

```
db2ncrt /n:node_number
        /u:username,password
        /p:logical_port
```

- /n:
The unique database partition 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 database partition on a computer. If you create a logical database partition, you must specify this parameter and select a logical port number that is not in use. There are several restrictions:

- On every computer 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 database partition). 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 computer 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 computer.

- /i:instance_name
The instance name; the default is the current instance.
- /m:computer_name
The computer name of the Windows workstation on which the database partition resides; the default name is the computer name of the local computer.
- /o:instance_owning_computer
The computer name of the computer that is the instance-owning computer; the default is the local computer. This parameter is required when the **db2ncrt** command is invoked on any computer that is not the instance-owning computer.

For example, if you want to add a new database partition server to the instance TESTMPP (so that you are running multiple logical database partitions) on the instance-owning computer MYMACHIN, and you want this new database partition to be known as database partition 2 using logical port 1, enter:

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
/M:TEST /o:MYMACHIN
```

Related reference:

- “db2icrt - Create instance command” in *Command Reference*
- “db2ncrt - Add database partition server to an instance command” in *Command Reference*
- “db2start - Start DB2 command” in *Command Reference*

Changing the database partition (Windows)

Procedure:

On Windows, use the **db2nchg** command to do the following:

- Move the database partition from one computer to another.
- Change the TCP/IP host name of the computer.
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.

The command has the following required parameter:

```
db2nchg /n:node_number
```

The parameter /n: is the 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 database 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 computer. 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:computer_name

Moves the database partition server to another computer. 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 computer 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 database partition 2, which participates in the instance TESTMPP, to use the logical port 3, enter the following command:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

The DB2 database manager provides the capability of accessing DB2 database system registry variables at the instance level on a remote computer. Currently, DB2 database system registry variables are stored in three different levels: computer or global level, instance level, and database partition level. The registry variables stored at the instance level (including the database partition level) can be redirected to another computer by using DB2REMOTEPREG. When DB2REMOTEPREG is set, the DB2 database manager will access the DB2 database system registry variables from the computer 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 database instance profiles and instance listings will be located on the specified remote computer name.
- If your environment includes users from domains, ensure that the logon account associated with the DB2 instance service is a domain account. This ensures that the DB2 instance has the appropriate privileges to enumerate groups at the domain level.

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

Related concepts:

- “DB2 registry and environment variables” in *Performance Guide*

Related reference:

- “db2nchg - Change database partition server configuration command” in *Command Reference*

Dropping a database partition from an instance (Windows)

Procedure:

On Windows, use the **db2ndrop** command to drop a database partition server from an instance that has no databases. If you drop a database partition server, its database partition 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 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 might cause inconsistencies in the partitioned database environment.

If you want to drop a database partition that is assigned the logical port 0 from a computer that is running multiple logical database partitions, you must drop all the other database partitions assigned to the other logical ports before you can drop the database partition assigned to logical port 0. Each database partition server must have a database partition assigned to logical port 0.

The command has the following parameters:

```
db2ndrop /n:node_number /i:instance_name
```

- **/n:**

The unique database partition 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 database partition zero (0) represents the instance-owning computer.

- **/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 *Performance Guide*

Related reference:

- “db2idrop - Remove instance command” in *Command Reference*
- “db2ndrop - Drop database partition server from an instance command” in *Command Reference*
- “db2stop - Stop DB2 command” in *Command Reference*

Creating table spaces

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

Table spaces

It is easier to manage very large databases if you partition them into separately managed parts called *table spaces*.

A table space lets you assign the location of data to particular logical devices or portions thereof. For example, when creating a table you can specify that its indexes or its long columns with long or large object (LOB) data be kept away from the rest of the table data.

A table space can be spread over one or more physical storage devices (containers) for increased performance. However, it is recommended that all the devices or containers within a table space have similar performance characteristics.

A table space can be managed in two different ways: as a system-managed space (SMS) or as a database-managed space (DMS).

Related concepts:

- “Container” on page 455

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.

If you do use the **CREATE DATABASE** command, you can specify the page size for the default buffer pool and the initial table spaces. This default also represents the default page size for all future **CREATE BUFFERPOOL** and **CREATE TABLESPACE** statements. If you do not specify the page size when creating the database, the default page size is 4 KB.

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 might 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 database 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:

- “System catalog tables” on page 175
- “Table space design” in *Administration Guide: Planning*

Related tasks:

- “Creating a table space” on page 149

Related reference:

- “CREATE DATABASE command” in *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.

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.

When you create a database, three initial table spaces are created. The page size for the three initial table spaces is based on the default that is established or accepted when you use the **CREATE DATABASE** command. This default also represents the default page size for all future **CREATE BUFFERPOOL** and **CREATE TABLESPACE** statements. If you do not specify the page size when creating the database, the default page size is 4 KB. If you do not specify the page size when creating a table space, the default page size is the one set when you created the database.

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:

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.

When creating table space containers, the database manager creates any directory levels that do not exist. For example, if a container is specified as

/project/user_data/container1, and the directory /project does not exist, then the database manager creates the directories /project and /project/user_data.

Starting with DB2 Universal Database Version 8.2, FixPak 4, any directories created by the database manager are created with PERMISSION 700. This means that only the owner has read, write, and execute access.

When creating multiple instances, note the following scenario:

1. Using the same directory structure as above, suppose that directory levels /project/user_data do not exist.
2. user1 creates an instance, named user1 by default, then creates a database, and then creates a table space with /project/user_data/container1 as one of its containers.
3. user2 creates an instance, named user2 by default, then creates a database, and then attempts to create a table space with /project/user_data/container2 as one of its containers.

Because the database manager created directory levels /project/user_data with PERMISSION 700 from the first request, user2 does not have access to these directory levels and cannot create container2 in those directories. In this case, the CREATE TABLESPACE operation fails.

There are two methods to resolve this conflict:

1. Create the directory /project/user_data before creating the table spaces and set the permission to whatever access is needed for both user1 and user2 to create the table spaces. If all levels of table space directory exist, the database manager does not modify the access.
2. After user1 creates /project/user_data/container1, set the permission of /project/user_data to whatever access is needed for user2 to create the table space.

If a subdirectory is created by the database manager, it might also be deleted by the database manager when the table space is dropped.

The assumption in the previous 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 Linux and UNIX 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/rdb1v6' 10000,
        DEVICE '/dev/rdb1v7' 10000,
        DEVICE '/dev/rdb1v8' 10000)
  OVERHEAD 7.5
  TRANSFERRATE 0.06
```

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 multi-partition 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 for new table spaces in databases created in DB2 Version 9.1 or later are:

- OVERHEAD 7.5 ms
- TRANSFERRATE 0.06 ms

New table spaces in databases created in earlier versions of DB2 use the following defaults:

- OVERHEAD 12.67 ms
- TRANSFERRATE 0.18 ms

DB2 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 Linux and UNIX system with an 8 KB page size.

```
CREATE TABLESPACE SMS8K
PAGE SIZE 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 following the ALTER TABLESPACE SQL statement to prevent system catalog contention.

Note: The PREFETCHSIZE value should be a multiple of the EXTENTSIZE value. 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:

$$\text{prefetch size} = (\text{number of containers}) \times (\text{number of physical spindles per container}) \times \text{extent size}$$

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

Direct I/O (DIO) improves memory performance because it bypasses caching at the file system level. This process reduces CPU overhead and makes more memory available to the database instance.

Concurrent I/O (CIO) includes the advantages of DIO and also relieves the serialization of write accesses.

DIO and CIO are supported on AIX; DIO is supported on HP-UX, Solaris Operating Environment, Linux, and Windows operating systems.

The keywords NO FILE SYSTEM CACHING and FILE SYSTEM CACHING are part of the CREATE and ALTER TABLESPACE SQL statements to allow you to specify whether DIO or CIO is to be used with each table space. When NO FILE SYSTEM CACHING is in effect, the database manager attempts to use Concurrent I/O (CIO) wherever possible. In cases where CIO is not supported (for example, if JFS is used), DIO is used instead.

When you issue the CREATE TABLESPACE statement, the dropped table recovery feature is turned on by default. This feature lets you recover dropped table data using table space-level restore and rollforward operations. This is useful because it is faster than database-level recovery, and your database can remain available to users.

However, the dropped table recovery feature can have some performance impact on forward recovery when there are many drop table operations to recover or when the history file is very large. You might want to disable this feature if you plan to run numerous drop table operations, and you either uses circular logging or you do not think you will want to recover any of the dropped tables. To disable this feature, you can explicitly set the DROPPED TABLE RECOVERY option to OFF when you issue the CREATE TABLESPACE statement. Alternatively, you can turn off the dropped table recovery feature for an existing table space using the ALTER TABLESPACE statement.

Related concepts:

- “Table space design” in *Administration Guide: Planning*
- “Table spaces in database partition groups” on page 163
- “Database managed space” in *Administration Guide: Planning*
- “System managed space” in *Administration Guide: Planning*
- “Sequential prefetching” in *Performance Guide*

Related tasks:

- “Recovering a dropped table” in *Data Recovery and High Availability Guide and Reference*
- “Enabling large page support in a 64-bit environment (AIX)” on page 12

Related reference:

- “ALTER TABLESPACE statement” in *SQL Reference, Volume 2*
- “CREATE TABLESPACE statement” in *SQL Reference, Volume 2*

Automatic resizing of table spaces

Two base table space types exist within the DB2 database system: system-managed space (SMS) and database-managed space (DMS).

The containers associated with SMS table spaces are file system directories and the files within these directories grow as the objects in the table space grow. The files grow until a file system limit has been reached on one of the containers or until the database’s table space size limit is reached (see SQL and XQuery limits).

DMS table spaces are made up of file containers or raw device containers, and their sizes are set when the containers are assigned to the table space. The table space is considered full when all of the space within the containers has been used. However, unlike SMS, you can add or extend containers using the ALTER TABLESPACE statement, allowing more storage space to be given to the table space. DMS table spaces also have a feature called “auto-resize”. As space is consumed in a DMS table space that can be automatically resized, the DB2 database system might extend one or more file containers. SMS table spaces have similar capabilities for growing automatically but the term “auto-resize” is used exclusively for DMS.

Enabling and Disabling Auto-Resize (AUTORESIZE):

By default, the auto-resize feature is not enabled for a DMS table space. The following statement creates a DMS table space that does not have auto-resize enabled:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
```

To enable the auto-resize feature specify the AUTORESIZE YES clause as part of the CREATE TABLESPACE statement:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M) AUTORESIZE YES
```

You can also enable or disable the auto-resize feature after a DMS table space has been created by using the AUTORESIZE clause on the ALTER TABLESPACE statement:

```
ALTER TABLESPACE DMS1 AUTORESIZE YES
ALTER TABLESPACE DMS1 AUTORESIZE NO
```

Two other attributes, MAXSIZE and INCREASESIZE, are associated with auto-resize table spaces.

Maximum size (MAXSIZE):

The MAXSIZE clause on the CREATE TABLESPACE statement defines the maximum size for the table space. For example, the following statement creates a table space that can grow to 100 megabytes (per database partition if the database has multiple database partitions):

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES MAXSIZE 100 M
```

The MAXSIZE NONE clause specifies that there is no maximum limit for the table space. The table space can grow until a file system limit or until the DB2 table space limit has been reached (see the SQL Limits section in the *SQL Reference*). No maximum limit is the default if the MAXSIZE clause is not specified when the auto-resize feature is enabled.

The ALTER TABLESPACE statement changes the value of MAXSIZE for a table space that has auto-resize already enabled. For example:

```
ALTER TABLESPACE DMS1 MAXSIZE 1 G
ALTER TABLESPACE DMS1 MAXSIZE NONE
```

If a maximum size is specified, the actual value that DB2 enforces might be slightly smaller than the value provided because DB2 attempts to keep container growth consistent. It might not be possible to extend the containers by equal amounts and reach the maximum size exactly.

Increase size (INCREASESIZE):

The INCREASESIZE clause on the CREATE TABLESPACE statement defines the amount of space used to increase the table space when there are no free extents within the table space, and a request for one or more extents has been made. The value can be specified as an explicit size or as a percentage. For example:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 5 M

CREATE TABLESPACE DMS1 MANAGED BY DATABASE
  USING (FILE '/db2files/DMS1' 10 M)
  AUTORESIZE YES INCREASESIZE 50 PERCENT
```

A percentage value means that the increase size is calculated every time that the table space needs to grow, and growth is based on a percentage of the table space size at that time. For example, if the table space is 20 megabytes in size and the increase size is 50 percent, the table space grows by 10 megabytes the first time (to a size of 30 megabytes) and by 15 megabytes the next time.

If the INCREASESIZE clause is not specified when the auto-resize feature is enabled, DB2 determines an appropriate value to use, which might change over the life of the table space. Like AUTORESIZE and MAXSIZE, you can change the value of INCREASESIZE using the ALTER TABLESPACE statement.

If a size increase is specified, the actual value used by DB2 might be slightly different than the value provided. This adjustment in the value used is done to keep growth consistent across the containers in the table space.

How table spaces are extended:

For table spaces that can be automatically resized, DB2 attempts to increase the size of the table space when all of the existing space has been used and a request

for more space is made. DB2 determines which of the containers can be extended in the table space so that a rebalance does not occur. DB2 extends only those containers that exist within the last range of the table space map (the map describes the storage layout for the table space), and they are all extended by an equal amount.

For example, consider the following statement:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
  USING (FILE 'C:\TS1CONT' 1000, FILE 'D:\TS1CONT' 1000,
        FILE 'E:\TS1CONT' 2000, FILE 'F:\TS1CONT' 2000)
  EXTENTSIZE 4
  AUTORESIZE YES
```

Keeping in mind that DB2 uses a small portion (one extent) of each container for meta-data, here is the table space map that is created for the table space based on the CREATE TABLESPACE statement. (The table space map is part of the output from a table space snapshot).

Table space map:

Range Number	Stripe Set	Stripe Offset	Max Extent	Max Page	Start Stripe	End Stripe	Adj.	Containers
[0]	[0]	0	995	3983	0	248	0	4 (0,1,2,3)
[1]	[0]	0	1495	5983	249	498	0	2 (2,3)

The table space map shows that the containers with an identifier of 2 and 3 (E:\TS1CONT and F:\TS1CONT) are the only containers in the last range of the map. Therefore, when DB2 automatically extends the containers in this table space, it will extend only those two containers.

Note: If a table space is created with all of the containers having the same size, there is only one range in the map. In such a case, DB2 extends each of the containers. To prevent restricting extensions to only a subset of the containers, create a table space with containers of equal size.

As discussed in the MAXSIZE section, a limit on the maximum size of the table space can be specified, or a value of NONE can be provided, which allows for no limit on the growth. (When NONE or no limit is used, the upper limit is actually defined by the file system limit or by the DB2 table space limit.) DB2 does not attempt to increase the table space past the upper limit. However, before that limit is reached, an attempt to increase a container might fail due to a full file system. In this case, DB2 does not increase the table space any further and an “out of space” condition will be returned to the application.

There are two ways to resolve this situation:

- Increase the amount of space available on the file system that is full.
- Perform container operations against the table space such that the container in question is no longer in the last range of the table space map. The easiest way to make the container in question no longer the last in the range of the table space map is to add a new stripe set to the table space with a new set of containers. And the best practice is to ensure that the containers are all the same size. New stripe sets can be added using the BEGIN NEW STRIPE SET clause of the ALTER TABLESPACE statement. By adding new stripe sets, new ranges are added to the table space map. With new ranges, the containers that DB2 automatically attempts to extend are within this new stripe set and the older containers remain unchanged.

Note: When a user-initiated container operation is pending or a subsequent rebalance is in progress, the automatic resizing feature is disabled until the operation is committed or the rebalance is complete.

For example, a table space has three containers that are the same size and each resides on its own file system. As work is done against the table space, DB2 automatically extends these three containers. Eventually, one of the file systems becomes full, and the corresponding container can no longer grow. If more free space cannot be made available on the file system you must perform container operations against the table space such that the container in question is no longer in the last range of the table space map. In this case, you could add a new stripe set specifying two containers (one on each of the file systems that still has space), or you could specify more or fewer containers (again, making sure that each container being added is the same size and that there is sufficient room for growth on each of the file systems being used). When DB2 attempts to increase the size of the table space, it will now attempt to extend the containers in this new stripe set instead of the older containers.

The situation described above only applies to automatic storage table spaces that are not enabled for automatic resizing. If an automatic storage table space is enabled for automatic resizing, DB2 handles the full file system condition automatically by adding a new stripe set of containers.

Monitoring:

Automatic resizing for DMS table spaces is displayed as part of the table space monitor snapshot output. The increase size and maximum size values are also displayed:

Auto-resize enabled	= Yes or No
Current tablespace size (bytes)	= ###
Maximum tablespace size (bytes)	= ### or NONE
Increase size (bytes)	= ###
Increase size (percent)	= ###
Time of last successful resize	= DD/MM/YYYY HH:MM:SS.SSSSSS
Last resize attempt failed	= Yes or No

Usage notes:

Automatic resizing with table spaces has the following implications:

- Table spaces that are enabled for automatic resizing have meta-data associated with them that is not recognized by DB2 Universal Database Version 8.2.1 or earlier. Any attempt to use a database with table spaces enabled for automatic resizing on these versions results in a failure (most likely returning an SQL0980C or SQL0902C error). An error might be sent for trying to connect to a database or trying to restore a database. If table spaces are enabled for automatic resizing, disabling the “auto-resize” feature for these table spaces removes the meta-data, allowing the database to be used on DB2 Version 8.2.1 or earlier.
- When disabling the “auto-resize” feature, the values that are associated with INCREASESIZE and MAXSIZE are lost if this feature is subsequently enabled.
- This feature cannot be enabled for table spaces that use raw device containers. Also, raw device containers cannot be added to a table space that can be automatically resized. These operations result in errors (SQL0109N). If you need to add raw device containers, you must disable the feature first.
- A redirected restore operation cannot change the container definitions to include a raw device container (SQL0109N).

- Because the maximum size limits how DB2 automatically increases a table space, the maximum size also limits how users can increase a table space. In other words, when performing an operation that adds space to a table space, the resulting size must be less than or equal to the maximum size. Space can be added using the ADD, EXTEND, RESIZE, or BEGIN NEW STRIPE SET clauses of the ALTER TABLESPACE statement.

Related concepts:

- “How containers are added and extended in DMS table spaces” in *Administration Guide: Planning*
- “Table space maps” in *Administration Guide: Planning*
- “Automatic storage databases” on page 54

Related reference:

- “ALTER TABLESPACE statement” in *SQL Reference, Volume 2*
- “CREATE TABLESPACE statement” in *SQL Reference, Volume 2*

Creating a system temporary table space

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

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”.

Prerequisites:

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

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:

To create another system temporary table space, use the CREATE TABLESPACE statement. 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 159

Related reference:

- “CREATE TABLESPACE statement” in *SQL Reference, Volume 2*

Creating a user temporary table space

User temporary table spaces are not created by default when a database is created. If your application programs need to use temporary tables, you need to create a user temporary table space where the temporary tables will reside.

Like regular table spaces, user temporary table spaces can be created in any database partition group other than IBMTEMPGROUP. IBMDEFAULTGROUP is the default database partition group that is used when creating a user temporary table.

The DECLARE GLOBAL TEMPORARY TABLE statement defines declared temporary tables for use within a user temporary table space.

Procedure:

To create a user temporary table space, use the CREATE TABLESPACE statement:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
MANAGED BY DATABASE
USING (FILE 'd:\db2data\user_tbsp' 5000,
FILE 'e:\db2data\user_tbsp' 5000)
```

Related tasks:

- “Creating a user-defined temporary table” on page 212

Related reference:

- “CREATE TABLESPACE statement” in *SQL Reference, Volume 2*
- “DECLARE GLOBAL TEMPORARY TABLE statement” in *SQL Reference, Volume 2*

Creating table spaces without file system caching

The operating system, by default, caches file data that is read from and written to disk. A typical read operation involves physical disk access to read the data from disk into the file system cache, and then to copy the data from the cache to the application buffer. Similarly, a write operation involves physical disk access to copy the data from the application buffer into the file system cache, and then to copy it from the cache to the physical disk. This default behavior of caching data at the file system level is reflected in the DB2 table space clause: FILE SYSTEM CACHING with the default value of “Yes”. Since the DB2 database manager manages its own data caching using buffer pools, the caching at the file system level is not needed if the size of the buffer pool is tuned appropriately. In some cases, caching at the file system level and in the DB2 buffer pools causes performance degradation because of the extra CPU cycles required to do the double caching.

To avoid this double caching, most file systems have a feature that disables caching at the file system level. This is generically referred to as *non-buffered I/O*. On UNIX, this feature is commonly known as *Direct I/O (or DIO)*. On Windows, this is equivalent to opening the file with the FILE_FLAG_NO_BUFFERING flag. In addition, some file systems such as IBM JFS2 or VERITAS VxFS also support enhanced Direct I/O, that is, the higher-performing *Concurrent I/O (CIO)* feature. The DB2 database manager automatically takes advantage of CIO on file systems

where this feature exists. These features might help to reduce the memory requirements of the file system cache, thus making more memory available for other uses.

As stated above, the DB2 database manager automatically enables file system caching when performing I/O. To disable it, you can use the **CREATE TABLESPACE** or **ALTER TABLESPACE** statements. Use the **NO FILE SYSTEM CACHING** clause to enable non-buffered I/O, thus disabling file caching for a particular table space. Once enabled, the DB2 database manager automatically determines which of the DIO or CIO is to be used on all platforms. Given the performance improvement in CIO, the DB2 database manager uses it whenever it is supported; there is no user-interface to specify which one is to be used.

In order to obtain the maximum benefits of non-buffered I/O, it might be necessary to increase the size of DB2 buffer pools to mitigate any loss of benefit from file caching.

Prerequisites:

Table 16 shows the supported configuration for using table spaces without file system caching. It also indicates whether DIO or enhance DIO will be used in each case.

Table 16. Supported configuration for table spaces without file system caching.

Platforms	File system type and minimum level required	DIO or CIO requests submitted by the DB2 database manager
AIX 5.2+	Journal File System (JFS)	DIO
AIX 5.2+	Concurrent Journal File System (JFS2)	CIO
AIX 5.2+	VERITAS Storage Foundation for DB2 4.0 (VxFS)	CIO
HP-UX 11i (PA-RISC)	VERITAS Storage Foundation for DB2 3.5 (VxFS)	DIO
HP-UX Version 11i v2 (Itanium)	VERITAS Storage Foundation for DB2 3.5 (VxFS)	DIO
Solaris 9	UNIX File System (UFS)	DIO
Solaris 10	UNIX File System (UFS)	CIO
Solaris 9, 10	VERITAS Storage Foundation for DB2 4.1 (VxFS)	CIO
Linux distributions SLES 9 and RHEL 4 (on these architectures) x86, x86_64, IA64, POWER	ext2, ext3, reiserfs	DIO
Linux distributions SLES 9 and RHEL 4 (on these architectures) x86, x86_64, IA64, POWER	VERITAS Storage Foundation 4.1 (VxFS)	CIO

Table 16. Supported configuration for table spaces without file system caching. (continued)

Platforms	File system type and minimum level required	DIO or CIO requests submitted by the DB2 database manager
Linux distributions SLES 9 and RHEL 4 (on this architecture) zSeries	ext2, ext3 or reiserfs on a Small Computer System Interface (SCSI) disks using Fibre Channel Protocol (FCP)	DIO
Windows	No specific requirement, works on all DB2 supported file systems	DIO

Note: The VERITAS Storage Foundation for the DB2 database manager might have different operating system prerequisites. The platforms listed above is the prerequisite for the current DB2 release. Consult the VERITAS Storage Foundation for DB2 support information for the prerequisite information.

Procedure:

The recommended method of enabling non-buffered I/O is at the table space level, using the DB2 implementation method. This method allows you to apply non-buffered I/O on specific table spaces while avoiding any dependency on the physical layout of the database. It also allows the DB2 database manager to determine which I/O is best used for each file, buffered or non-buffered.

The clauses `NO FILE SYSTEM CACHING` and `FILE SYSTEM CACHING` can be specified in the `CREATE` and `ALTER TABLESPACE` statements to disable or enable file system caching, respectively. The default is `FILE SYSTEM CACHING`. In the case of `ALTER TABLESPACE`, existing connections to the database must be terminated before the new caching policy takes effect.

Example 1: `CREATE TABLESPACE <table space name>...`

By default, this new table space will be created using buffered I/O; the `FILE SYSTEM CACHING` clause is implied.

Example 2: `CREATE TABLESPACE <table space name> ... NO FILE SYSTEM CACHING`

The new `NO FILE SYSTEM CACHING` clause indicates that file system level caching will be OFF for this particular table space.

Example 3: `ALTER TABLESPACE <table space name> ... NO FILE SYSTEM CACHING`

This statement disables file system level caching for an existing table space.

Example 4: `ALTER TABLESPACE <table space name> ... FILE SYSTEM CACHING`

This statement enables file system level caching for an existing table space.

This method of disabling file system caching provides control of the I/O mode, buffered or non-buffered, at the table space level. Note that I/O access to Long Field (LF) and Large Objects (LOBs) will be buffered for both SMS and DMS containers.

The **GET SNAPSHOT FOR TABLESPACES** command can be used to query the current setting of the file system caching clause. For example, the following is a snippet from the DB2 GET SNAPSHOT FOR TABLESPACES ON db1 output:

```
Tablespace name           = USERSPACE1
Tablespace ID             = 2
Tablespace Type           = Database managed space
Tablespace Content Type   = All permanent data. Large table space.
Tablespace Page size (bytes) = 4096
Tablespace Extent size (pages) = 32
Automatic Prefetch size enabled = Yes
Buffer pool ID currently in use = 1
Buffer pool ID next startup = 1
Using automatic storage = Yes
Auto-resize enabled = Yes
File system caching = Yes
Tablespace State = 0x'00000000'
Detailed explanation:
  Normal
Tablespace Prefetch size (pages) = 32
Total number of pages = 256
```

Alternate methods to enable non-buffered I/O on UNIX:

Some UNIX platforms support the disabling of file system caching at a file system level by using the MOUNT option. Consult your operating system documentation for more information. However, it is important to understand the difference between disabling file system caching at the table space level and at the file system level. At the table space level, the DB2 database manager controls which files are to be opened with and without file system caching. At the file system level, every file residing on that particular file system will be opened without file system caching. Some platforms such as AIX have certain requirements before you can use this feature, such as serialization of read and write access. While the DB2 database manager adheres to these requirements, if the target file system contains non-DB2 files, before enabling this feature, consult your operating system documentation for any requirements.

In DB2 Version 8.1 FixPak 4, the registry variable DB2_DIRECT_IO disables file system caching for all SMS containers except for Long Field (LF), Large Objects (LOBs), and temporary table spaces on AIX JFS2. With the ability to enable this feature at the table space level, starting in DB2 Version 8.2, this registry variable has been deprecated. Setting this registry variable in DB2 Version 9.1 is equivalent to altering all table spaces, SMS and DMS, with the NO FILE SYSTEM CACHING clause.

Alternate methods to enable non-buffered I/O on Windows:

In previous DB2 releases, the performance registry variable DB2NTNOCACHE could be used to disable file system caching for all DB2 files. This option remains available. The difference between DB2NTNOCACHE and using the NO FILE SYSTEM CACHING clause is the ability to disable caching for selective table spaces.

Related concepts:

- “Buffer pool management” in *Performance Guide*

Related reference:

- “ALTER TABLESPACE statement” in *SQL Reference, Volume 2*
- “CREATE TABLESPACE statement” in *SQL Reference, Volume 2*

- “fs_caching - File System Caching monitor element” in *System Monitor Guide and Reference*

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 database 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 *SQL Reference, Volume 2*

Attaching a direct disk access device

When working with containers to store data, the DB2 database manager supports direct disk access (raw I/O). This type of support allows you to attach a direct disk access (raw) device to any DB2 database 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 physical and logical methods for identifying direct disk access differs based on operating system:

- On the Windows operating systems::

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
```

To specify a logical drive, that is, an unformatted database 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.

For Windows, there is a new method for specifying DMS raw table space containers. Volumes (that is, basic disk database 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 definition. The GUIDs are unique across systems which means that in a multi-partition database, GUIDs are different for each database partition even if the disk partition definitions 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.

- On Linux and UNIX 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 database 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*.

Related tasks:

- “Setting up a direct disk access device on Linux” on page 164

Setting up a direct disk access device on Linux

When working with containers to store data, DB2 supports direct disk (raw) access using the block device interface (that is, raw I/O). The following information should be used when working in a Linux environment.

Prerequisites:

Before setting up raw I/O on Linux, one or more free IDE or SCSI disk database partitions are required.

In order to reference the disk partition when creating the table space, you must know the name of the disk partition and the amount of space associated with the disk partition that is to be allocated to the table space.

Restrictions:

Raw devices are not supported by DB2 on Linux/390.

Procedure:

To configure raw I/O on Linux:

In this example, the raw database partition to be used is /dev/sda5. It should not contain any valuable data.

1. Calculate the number of 4 096-byte pages in this database 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 17. 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( (4682947 * 1024)/4096 )
num_pages = 1170736
```

2. Create the table space in DB2 by specifying the disk partition name. For example:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/sda5' 1170736)
```

3. To specify logical partitions by using junction points (or volume mount points), mount the RAW partition to another NTFS-formatted volume as a junction point, then specify the path to the junction point on the NTFS volume as the container path. For example:

```
CREATE TABLESPACE TS4
MANAGED BY DATABASE USING (DEVICE 'C:\JUNCTION\DISK_1' 10000,
DEVICE 'C:\JUNCTION\DISK_2' 10000)
```

DB2 first queries the partition to see whether there is a file system on it; if yes, the partition is not treated as a RAW device, and DB2 performs normal file system I/O operations on the partition.

Table spaces on raw devices are also supported for all other page sizes supported by the DB2 database manager.

Prior to Version 9, direct disk access using a raw controller utility on Linux was used. This method has been deprecated by the operating system, and it's use is

discouraged. DB2 will still allow you to use this method if the Linux operating system still supports it, however, there will be a message in the db2diag.log that will indicate that its use is deprecated.

The prior method would have required you to "bind" a disk partition to a raw controller, then specify that raw controller to DB2 using the **CREATE TABLESPACE** command:

```
CREATE TABLESPACE dms1
  MANAGED BY DATABASE
  USING (DEVICE '/dev/raw/raw1' 1170736)
```

Related tasks:

- "Attaching a direct disk access device" on page 163

Creating a buffer pool

When you create a database, a default buffer pool is created. The page size for the default buffer pool is set when you use the **CREATE DATABASE** command. This default represents the default page size for all future **CREATE BUFFERPOOL** and **CREATE TABLESPACE** statements. If you do not specify the page size when creating the database, the default page size is 4 KB. If you do not specify the page size when creating a buffer pool, the default page size is the one set when you created the database.

However, you might need a buffer pool that has different characteristics than the default buffer pool. You can create new buffer pools for the database manager to use. Buffer pools improve database system performance immediately.

The page sizes that you specify 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 do you want to use?
- Will the buffer pool be created immediately or following the next time that the database is deactivated and reactivated?
- Do you want to associate the buffer pool with a subset of all database partitions that make up the database?
- What page size do you want to specify for the buffer pool?
- Will you specify a fixed size for the buffer pool, or will you allow DB2 to adjust the size of the buffer pool in response to the requirements of your workload? It is recommended that you allow DB2 to tune your buffer pool automatically by leaving the size parameter unspecified during buffer pool creation

Procedure:

To create a buffer pool using the Control Center:

1. Open the Create Buffer Pool window: From the Control Center, expand the object tree until you find the **Buffer Pools** folder. Right-click the **Buffer Pools** folder and select **Create** from the pop-up menu. The Create Buffer Pool window opens.
2. Type a new name for the buffer pool.
3. Specify the size of the pages to be used for the buffer pool. The valid values are 4 KB, 8 KB, 16 KB, and 32 KB.
4. Type the size of the buffer pool in pages.
5. Specify whether to use the default buffer pool size.
6. Specify whether to create the buffer pool immediately (this is the default setting), or whether to create it the next time that the database is restarted.

To create a buffer pool using the command line:

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 concepts:

- "Self tuning memory" in *Performance Guide*

Related tasks:

- "Altering a buffer pool" on page 283

Related reference:

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

Creating buffer pools for partitioned databases

From the Control Center, use the Create Buffer Pool window to create new buffer pools for partitioned databases. If you choose to create the buffer pool when you restart the database, any new tables or table spaces will use the default buffer pool. Any changes to a table space that specifies this buffer pool will continue to use its existing buffer pool.

Prerequisites:

To create or alter a buffer pool, you must have either SYSADM or SYSCTRL authority.

Procedure:

1. Open the Create Buffer Pool window: From the Control Center, expand the object tree until you find the **Buffer Pools** folder. Right-click the **Buffer Pools** folder and select **Create** from the pop-up menu. The Create Buffer Pool window opens.
2. Type a new name for the buffer pool.

3. Specify the size of the pages to be used for the buffer pool. The valid values are 4 KB, 8 KB, 16 KB, and 32 KB.
4. Specify when to create the buffer pool: immediately or the next time that the database is restarted.
5. Specify on which database partitions you want the buffer pool created.
6. Optional: Modify the default size of the buffer pool on selected database partitions.
7. Define the size of the buffer pool by the number of pages.

Related tasks:

- “Altering a buffer pool” on page 283
- “Creating a buffer pool” on page 166

Creating schemas

Schemas are used to organize object ownership within the database.

Creating a schema

While organizing your data into tables, it might 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.

Schemas might 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.

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 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.

Prerequisites:

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

To issue the `CREATE SCHEMA` statement, you must have `DBADM` authority.

To create a schema with any valid name, you need `SYSADM` or `DBADM` authority.

Restrictions:

If users do not have IMPLICIT_SCHEMA or DBADM authority, the only schema they can create is one that has the same name as their own authorization ID.

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).

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:

- "Implicit schema authority (IMPLICIT_SCHEMA) considerations" on page 513
- "Grouping objects by schema" on page 6
- "Schema privileges" on page 514

Related tasks:

- "Setting a schema" on page 169

Related reference:

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

Setting a schema

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

Procedure:

To establish a default schema: Set the special register CURRENT_SCHEMA to the schema you want to use as the default. For example:

```
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

and XQuery 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 *SQL Reference, Volume 1*

Related reference:

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

Copying a schema

Use the ADMIN_COPY_SCHEMA procedure to copy a single schema within the same database or use the db2move utility with the -co COPY action to copy a single schema or multiple schemas from a source database to a target database. Most database objects from the source schema are copied to the target database under the new schema. The db2move utility and the ADMIN_COPY_SCHEMA procedure allow you to quickly make copies of a database schema. Once a model schema is established, you can use it as a template for creating new versions.

For more information on ADMIN_COPY_SCHEMA, see the ADMIN_COPY_SCHEMA reference information.

Restrictions:

- The db2move utility attempts to successfully copy all allowable schema objects with the exception of the following types:
 - table hierarchy
 - staging tables (not supported by the load utility in multiple partition database environments)
 - jars (Java routine archives)
 - nicknames
 - packages
 - view hierarchies
 - object privileges (All new objects are created with default authorizations)
 - statistics (New objects do not contain statistics information)
 - index extensions (user-defined structured type related)
 - user-defined structured types and their transform functions
- If an object of one of the unsupported types is detected in the source schema, an entry is logged to an error file, indicating that a unsupported object type is detected. The COPY operation will still succeed; this logged entry is meant to inform users of objects not copied by this operation.
- Objects that are not coupled with a schema such as table spaces, and event monitors, are not operated on during a copy schema operation.
- When copying a replicated table, the new copy of the table is not enabled for replication. The table is re-created as a regular table.
- The source database must be cataloged if it does not reside in the same instance as the target database.

- Using the ADMIN_COPY_SCHEMA procedure with the COPYNO option, places the table spaces wherein the target database object resides into backup pending state. The ADMIN_COPY_SCHEMA procedure issues a SET INTEGRITY statement to get the table out of the Set Integrity Pending state after the load operation completes. In situations where a table has referential constraints defined, the table is placed in the Set Integrity Pending state. Because the table spaces are already in backup pending state, the ADMIN_COPY_SCHEMA procedure's attempt to issue a SET INTEGRITY statement will fail.

To resolve this situation, issue a backup to get the affected table spaces out of backup pending state. Next, look at the statement_text column of the error table generated by the ADMIN_COPY_SCHEMA procedure to find a list of tables in the Set Integrity Pending state. This is demonstrated in the following example.

Example 1:

```
db2 "select substr(OBJECT_SCHEMA,1, 8)
as OBJECT_SCHEMA, substr(OBJECT_NAME,1, 15)
as OBJECT_NAME, SQLCODE, SQLSTATE, ERROR_TIMESTAMP, substr(DIAGTEXT,1, 80)
as DIAGTEXT, substr(STATEMENT,1, 80)
as STATEMENT from COPYERRSCH.COPYERRTAB"
```

```
OBJECT_SCHEMA OBJECT_NAME      SQLCODE      SQLSTATE ERROR_TIMESTAMP
-----
SALES          EXPLAIN_STREAM      -290 55039    2006-03-18-03.22.34.810346
```

DIAGTEXT

```
-----
[IBM][CLI Driver][DB2/LINUX8664] SQL0290N Table space access is not allowed.
```

STATEMENT

```
-----
set integrity for "SALES"."ADVISE_INDEX", "SALES"."ADVISE_MQT", "SALES"."
```

1 record(s) selected.

Finally, issue the SET INTEGRITY statement for each of the tables listed to take each table out of the Set Integrity Pending state.

Procedure:

This utility must be invoked on the target system if source and target schemas reside on different systems. For copying schemas from one database to another, this action requires a list of schema names to be copied from a source database, separated by commas, and a target database name.

To copy a schema using the command line processor (CLP), use the following syntax:

```
db2move <dbname> COPY -co <COPY-options>
-u <userid> -p <password>
```

Example 2:

The following is an example of a db2move -co COPY operation that copies schema BAR into FOO from the sample database to the target database:

```
db2move sample COPY -sn BAR -co target_db target schema_map
"((BAR,FOO))" -u userid -p password
```

The new (target) schema objects are created using the same object names as the objects in the source schema, but with the target schema qualifier. It is possible to

create copies of tables with or without the data from the source table. The source and target databases can be on different systems.

Example 3:

The following example shows you to specify specific table space name mappings to be used instead of the table spaces from the source system during a COPY operation. You can specify the SYS_ANY keyword to indicate that the target table space should be chosen using the default table space selection algorithm. In this case, the db2move tool chooses any available table space to be used as the target. For example:

```
db2move sample COPY -sn BAR -co target_db target schema_map
"((BAR,F00))" tablespace_map "(SYS_ANY)" -u userid -p password
```

The SYS_ANY keyword can be used for all table spaces, or you can specify specific mappings for some table spaces, and the default table space selection algorithm for the remaining. For example:

```
db2move sample COPY -sn BAR -co target_db target schema_map "
((BAR,F00))" tablespace_map "(TS1, TS2),(TS3, TS4), SYS_ANY)"
-u userid -p password
```

This indicates that table space TS1 is mapped to TS2, TS3 is mapped to TS4, but the remaining table spaces use a default table space selection algorithm.

Example 4:

You can also change the owner of each new object created in the target schema after a successful COPY. The default owner of the target objects is the connect user; if this option is specified, ownership is transferred to a new owner as demonstrated in the following example:

```
db2move sample COPY -sn BAR -co target_db target schema_map
"((BAR,F00))" tablespace_map "(SYS_ANY)" owner jrichards
-u userid -p password
```

The new owner of the target objects is jrichards.

Related concepts:

- “Schemas” in *SQL Reference, Volume 1*

Related tasks:

- “Dropping a schema” on page 294
- “Restarting a failed copy schema operation” on page 173
- “Setting a schema” on page 169

Related reference:

- “ADMIN_COPY_SCHEMA procedure – Copy a specific schema and its objects” in *Administrative SQL Routines and Views*
- “ADMIN_DROP_SCHEMA procedure – Drop a specific schema and its objects” in *Administrative SQL Routines and Views*
- “db2move - Database movement tool command” in *Command Reference*

Restarting a failed copy schema operation

Errors occurring during a db2move COPY operation can be handled in various ways depending on the type of object being copied, or the phase the COPY operation failure.

The db2move utility reports errors and messages to the user using message and error files. Copy schema operations use the COPYSHEMA_<timestamp>.MSG message file, and the COPYSHEMA_<timestamp>.err error file. These files are created in the current working directory. The current time is appended to the filename to ensure uniqueness of the files. It is up to the user to delete these message and error files when they are no longer required.

Note: It is possible to have multiple db2move instances running simultaneously. The COPY option does not return any SQLCODES. This is consistent with db2move behavior.

Object types:

The type of object being copied can be categorized as one of two types : physical objects and business objects.

A physical object refers to an object that physically reside in a container, such as tables, indexes and user-defined structured types. A business object refers to cataloged objects that do not reside in containers, such as views, user-defined structured types (UDTs), and aliases.

Copy schema errors relating to physical objects:

Failures which occur during the recreate of physical objects on the target database, are logged in the error file COPYSHEMA_<timestamp>.err. For each failing object, the error file contains information such as object name, object type, DDL text, time stamp, and a string formatted sqlca (sqlca field names, followed by their data values).

Example 1:

Sample output for the COPYSHEMA_<timestamp>.err error file:

```
1. schema: F00.T1
   Type:    TABLE
Error Msg: SQL0104N An unexpected token 'F00.T1'...
Timestamp: 2005-05-18-14.08.35.65
DDL:      create view F00.v1

2.schema:  F00.T3
   Type:    TABLE
Error Msg: SQL0204N F00.V1 is an undefined name.
Timestamp: 2005-05-18-14.08.35.68
DDL:      create table F00.T3
```

If any errors creating physical objects are logged at the end of the recreate phase and before attempting the load phase, the db2move utility fails and an error is returned. All object creation on the target database is rolled back, and all internally created tables are cleaned up on the source database. The rollback occurs at the end of the recreate phase after attempting to recreate each object, rather than after the first failure, in order to gather all possible errors into the error file. This allows

you the opportunity to fix any problems before restarting the db2move operation. If there are no failures, the error file is deleted.

Copy schema errors relating to business objects:

Failures that occur during the recreation of business objects on the target database, do not cause the db2move utility to fail. Instead, these failures are logged in the COPYSHEMA_<timestamp>.err error file. Upon completion of the db2move utility, you can examine the failures, address any issues, and manually recreate each failed object (the DDL is provided in the error file for convenience).

If an error occurs while db2move is attempting to repopulate table data using the load utility, the db2move utility will not fail. Rather, generic failure information is logged to the COPYSHEMA_<timestamp>.err file (object name, object type, DDL text, time stamp, sqlca, and so on), and the fully qualified name of the table is logged into another file, "LOADTABLE_<timestamp>.err". Each table is listed per line to satisfy the db2move -tf option format, similar to the following:

```
"F00"."TABLE1"  
"F00 1"."TAB 444"
```

Restarting the copy schema operation:

After addressing the issues causing the loads operations to fail (described in the error file), you can reissue the db2move -COPY command using the '-tf' option (passing in the LOADTABLE.err filename) as shown in the following syntax:

Example 2:

```
db2move sourcedb COPY -tf LOADTABLE.err -co TARGETDB mytargetdb  
-mode load_only
```

You can also input the table names manually using the -tn option, as shown in the following syntax:

Example 3:

```
db2move sourcedb COPY -tn "F00"."TABLE1","F00 1"."TAB 444",  
-co TARGETDB mytargetdb -mode load_only
```

Other types of failures:

Internal operations such as memory errors, or file system errors can cause the db2move utility to fail.

Should the internal operation failure occur during the ddl recreate phase, all successfully created objects are rolled back from the target schema, and all internally created tables such as the DMT table and the db2look table, are cleaned up on the source database.

Should the internal operation failure occur during the load phase, all successfully created objects remain on the target schema. All tables that experience a failure during a load operation, and all tables which have not yet been loaded are logged in the LOADTABLE.err error file. You can then issue the db2move COPY command using the LOADTABLE.err as discussed in Example 2. If the db2move utility abends (for example a system crash, the utility traps, the utility is killed, and so on), then the information regarding which tables still need to be loaded is

lost. In this case, you can drop the target schema using the `ADMIN_DROP_SCHEMA` procedure and reissue the `db2move COPY` command.

Regardless of what error you might encounter during an attempted copy schema operation, you always have the option of dropping the target schema using the `ADMIN_DROP_SCHEMA` procedure and reissuing the `db2move COPY` command.

Related reference:

- “`ADMIN_COPY_SCHEMA` procedure – Copy a specific schema and its objects” in *Administrative SQL Routines and Views*
- “`ADMIN_DROP_SCHEMA` procedure – Drop a specific schema and its objects” in *Administrative SQL Routines and Views*

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 might want to limit the access to the system catalog views.

Related concepts:

- “Catalog views” in *SQL Reference, Volume 1*
- “Functions overview” in *SQL Reference, Volume 1*
- “User-defined functions” in *SQL Reference, Volume 1*

Related tasks:

- “Securing the system catalog view” on page 613

Related reference:

- “Functions” in *SQL Reference, Volume 1*

Cataloging a database

When you create a new database, it is automatically cataloged in the system database directory file. You might 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.

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 might 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.

Prerequisites:

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

Procedure:

To catalog a database with a different alias name using the command line processor, use the **CATALOG DATABASE** command. For example, the following command line processor command catalogs the `person1` database as `humanres`:

```
CATALOG DATABASE person1 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 (`person1`).

To catalog a database in the system database directory from a client application, call the `sqlcadb` API.

To catalog a database on an instance other than the default using the command line processor, use the **CATALOG DATABASE** command. 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 computers.

Related tasks:

- “Updating the directories with information about remote database server computers” on page 180

Related reference:

- “CATALOG DATABASE command” in *Command Reference*
- “TERMINATE command” in *Command Reference*
- “dir_cache - Directory cache support configuration parameter” in *Performance Guide*
- “UNCATALOG DATABASE command” in *Command Reference*

Cataloging database systems

In order to access DB2 instances and databases on other servers, DB2 needs to catalog that system in the admin node directory of the client. Adding a system adds an entry to the admin node directory giving DB2 the information it needs to communicate with the remote system. All systems that you access will have an entry in the admin node directory.

Prerequisites:

To add or change a system, you must have SYSADM or SYSCTRL authority.

Procedure:

1. Open the Add System window: From the Control Center, right-click the All Systems folder and select **Add** from the pop-up menu. The Add System window opens.
2. To add a DB2 system, click the **DB2** radio button. Then do the following:
 - a. Specify the physical machine, server system, or workstation where the target database is located. The system name on the server system is defined by the DB2SYSTEM DAS configuration parameter. This is the value that you should use. If your network supports TCP/IP, then you can use discovery to help complete the remaining fields on this window.
 - b. Type the host name or IP (Internet Protocol) address where the target database resides. Issuing the TCP/IP hostname command on the server system retrieves the server’s host name. Issuing a ping hostname command will return the IP address of the host.
 - c. Specify a local nickname for the remote node where the database is located. The node name you choose must not already exist in the node directory or the admin node directory.
 - d. Specify the operating system where the target database is located.
 - e. Optional: Select **LDAP**, if the Lightweight Directory Access Protocol (LDAP) is enabled and you want to catalog the system in the LDAP directory.
3. To add an IMSplex, click the **IMS** radio button. Then do the following:
 - a. Specify the IMSplex name that you want to add. The name must match the identifier you have specified in the CSLSIxxx, CSLRIxxx, DFSCGxxx, and CSLOIxxx proclib members for the IMSPLEX= parameter.
 - b. Type the TCP/IP address of the IMSplex host and the port number that binds to the socket that IMS Connect manages. Valid port numbers are defined in the PORTID parameter of the HWSCFGxxx proclib member.
 - c. Select **OS/390** or **z/OS** as the operating system. This is the only option, if you are working with an IMSplex.

Related concepts:

- “About systems” in *Administration Guide: Planning*

Related reference:

- “CATALOG DATABASE command” in *Command Reference*

Database directories, directory services, and logs

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 *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, `sqlbdbir`, in the `sqlbdbir` 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:

- “Cataloging a database” on page 176
- “Enabling database partitioning in a database” on page 9

Related reference:

- “CREATE DATABASE command” in *Command Reference*

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 *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 computer.

Related reference:

- “CATALOG LOCAL NODE command” in *Command Reference*
- “CATALOG NAMED PIPE NODE command” in *Command Reference*

- “CATALOG TCPIP/TCPIP4/TCPIP6 NODE command” in *Command Reference*
- “LIST NODE DIRECTORY command” in *Command Reference*

Changing database directory information

You can use the Configuration Assistant to change the database directory information associated with a database.

Procedure:

To change database directory information using the Configuration Assistant:

1. Open the Change Database window or notebook: In the Configuration Assistant advanced view, click the Databases tab and select the database that you want to work with. From the **Selected** menu, click **Change Database**. The Change Database window opens, or if LDAP is enabled, the Change Database notebook opens.
Note: You cannot open the Change Database window or notebook unless you are in the Advanced view. If you are not in the Advanced view, you can use the Change Database wizard to change databases. To switch to the Advanced view, from the View menu select **Advanced View**.
2. Change fields as required. Clicking **OK** commits the changes for the selected database.

Related concepts:

- “Local database directory” on page 178
- “System database directory” on page 178

Related tasks:

- “Searching the LDAP servers” on page 585
- “Viewing the local or system database directory files” on page 179

Related reference:

- “LIST DATABASE DIRECTORY command” in *Command Reference*

Updating the directories with information about remote database server computers

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

Prerequisites:

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

Procedure:

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 7.1 (or later), or a DB2 UDB for z/OS Version 8 (or later), or a DB2 Universal Database for AS/400 Version 5.2 (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:

- “DCS directory values” in *DB2 Connect User’s Guide*
- “System database directory values” in *DB2 Connect User’s Guide*

Related reference:

- “CATALOG DATABASE command” in *Command Reference*
- “CATALOG DCS DATABASE command” in *Command Reference*
- “CATALOG TCPIP/TCPIP4/TCPIP6 NODE command” in *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 computer. Client applications search the LDAP directory for information required to connect to the database.

As an administrator of a DB2 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 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 domain environment, an LDAP server is already available because it is integrated with the Windows Active Directory. As a result, every computer running Windows can use LDAP.

An LDAP directory is helpful in an enterprise environment where it is difficult to update local directory catalogs on each client computer 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 107
- “Lightweight Directory Access Protocol (LDAP) overview” on page 573

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 might not have been physically written to disk, are redone. These actions ensure the integrity of the database.

Related concepts:

- “Understanding recovery logs” in *Data Recovery and High Availability Guide and Reference*

Administration notification log

When significant events occur, the DB2 database manager writes information to the administration notification log. For example, it records the status of DB2 utilities (REORG, BACKUP, RECOVERY, ROLL-FORWARD), high-level application issues, licensing activity, log file paths and storage problems, monitoring and indexing activities, table space problems, and so on. This information is intended for use by database and system administrators.

Notification messages provide additional information to supplement the SQLCODE that is provided. The type of event and the level of detail of the information gathered are determined by the NOTIFYLEVEL configuration parameter. This detailed diagnostic information is recorded in the db2diag.log text log file, not in the administration log. Diagnostic information is used for problem determination and is intended for DB2 customer support. The level of detail is determined by the DIAGLEVEL configuration parameter.

Related concepts:

- “Interpreting administration notification log file entries” in *Troubleshooting Guide*
- “Interpreting diagnostic log file entries” in *Troubleshooting Guide*
- “Interpreting the db2diag.log file informational record” in *Troubleshooting Guide*

Related tasks:

- “Setting the diagnostic log file error capture level” in *Troubleshooting Guide*
- “Setting the error capture level for the administration notification log file” in *Troubleshooting Guide*

Related reference:

- “notifylevel - Notify level configuration parameter” in *Performance Guide*
- “diaglevel - Diagnostic error capture level configuration parameter” in *Performance Guide*

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 and XQuery statements from a single source file.

Note: If you want to use these utilities from a client, you must bind them explicitly.

Procedure:

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 113

Related reference:

- “BIND command” in *Command Reference*

Generating DDL statements for database objects

Use the Generate DDL notebook to generate DDL (Data Definition Language) statements, SQL, and statistics in a script file to recreate database objects and their statistics in another database.

For information on generating DDL statements in DB2 for OS/390, see the DB2 for z/OS and OS/390 help.

To generate DDL for database objects, you need SELECT privilege on the system catalogs.

Opening the Generate DDL notebook:

- To open the Generate DDL notebook from a database, from the Control Center, expand the object tree until you find the database that contains the objects for which you want to create DDL. Right-click the database and click **Generate DDL** in the pop-up menu.
- To open the Generate DDL notebook from selected tables, from the Control Center, expand the object tree until you find the **Tables** folder of the database with which you are working. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window (the contents pane). Select the tables with which you want to work. Even if you are not going to generate DDL to recreate the tables, select the tables that relate to the objects for which you want to generate DDL. Right-click the selected tables, and click **Generate DDL** in the pop-up menu.
- To open the Generate DDL notebook from a selected schema, from the Control Center, expand the object tree until you find the **Schema** folder of the database with which you are working. Click the **Schema** folder. Any existing schemas are displayed in the pane on the right side of the window (the contents pane). Select the schema with which you want to work. Right click the schema and click **Generate DDL** in the pop-up menu.

From the Objects page, specify which statements you want to generate.

Specifying statement types:

On the Statement page and select the appropriate check boxes, as follows:

- **Database objects:** Generates DDL statements for the database objects, such as tables, indexes, views, triggers, aliases, UDFs, data types, and sequences, excluding any table spaces, database partition groups, and buffer pools that are user-defined.
- **Table spaces, database partition groups, and buffer pools:** Generates the DDL statements for these objects, excluding any of these objects that are user-defined.
- **Authorization statements:** Generates SQL authorization (GRANT) statements for the database objects.
- **Database statistics:** Generates SQL update statements for updating the statistics tables in the database.
- **Update statistics:** Only available if you have selected **Database statistics**. Generates the RUNSTATS command, which updates the statistics on the database that is generated.

Note: Choosing not to update the statistics allows you to create an empty database that the optimizer will treat as containing data.

- **Include COMMIT statements after every table:** Generates a COMMIT statement after the update statements for each table. The COMMIT statements are generated only when you select **Database statistics**.
- **Gather configuration parameters:** Gathers any configuration parameters and registry variables that are used by the SQL optimizer.
- **XML Schema Repository (XSR) objects: XML schemas, DTDs, external entities:** Generates statements to re-create XSR objects. If you select this check box, you must also specify the directory into which the generated XSR objects will be recreated.

Select the objects on which you want to base the generated DDL:

- If you accessed this notebook by selecting a table or multiple tables in the contents pane, then the object page displays all selected tables in a confirmation list. Clear the check boxes of any incorrect tables.
- If you accessed this notebook by selecting a database in the object tree, then the default is to generate DDL statements for all objects under the current user ID and schema within the database.
- If you accessed this notebook by selecting a schema in the contents pane, then the Schema field is prefilled with the name of the schema.

Limiting generation scope:

To limit the scope of the generation, and use the following options on the Object page:

- To limit DDL generation to objects created by a particular user, specify that user's ID in the **User** field.
- To limit the DDL generation to objects in a particular schema, specify that schema in the **Schema** field.
- To limit the DDL generation to objects related to specific tables, select the **Generate DDL for selected tables only** check box. Then, select the tables you want in the **Available tables** list box and move them to the **Selected tables** list box.

Run, save, and schedule options:

On the Schedule page, do one of the following:

- To run the task now, without creating a task in the Task Center or saving the task history to the Journal, select **Run now without saving task history**.
- To create a task for generating the DDL script and saving it in the Task Center, select **Create this as a task in the Task Center**. Then, specify the task information and options:
 - Specify the name of the system on which you want to run the task, in the Run system box. This system must be online at the time the task is scheduled to run.
 - Select the system where you want to store the task and the schedule information, in the Scheduler system drop-down box.
This system will store the task and notify the run system when it is time to run the task. The drop-down list contains any system that is cataloged and has the scheduler enabled. The scheduler system must be online so that it can notify the run system.
If the DB2 tools catalog is on a remote system, you will be asked for a user ID and password in order to connect to the database
 - Optional: If you want to select a different scheduling scheme, click **Advanced**. The Advanced Schedule Settings window opens where you can select between server scheduling or centralized scheduling.
 - To save the task in the Task Center, but not actually run the task, select **Save task only**.
 - To save the task in the Task Center and run the task now, select **Save and run task now**.
 - To save the task to the Task Center, and schedule a date and time to run the task later, specify **Schedule task execution**. The **Change** button is enabled.
Select a task in the table, and click Change. A window opens where you can enter the date and time that you want to run the task.

- The task table displays the task name suffix and the schedule information.
- To run a task in the Task Center you must supply a user ID and password. Type the user ID and password that you want to use to run the task.

You can use the Scheduler Settings page of the Tools Settings notebook to set the default scheduling scheme. Note that if you set a default scheduling scheme, you can still override it at the task level.

Optional: If you want to view the **db2look** command that is used to generate the DDL script, click **Show Command**.

Click **Generate** to generate DDL script. From the window that opens, you can do the following:

- Copy the script to the Command Editor
- Save the script to the file-system
- Run the script, and optionally save it to the Task Center.

Related concepts:

- “DDL Data definition language” in *SQL Guide*
- “Savepoints and Data Definition Language (DDL)” in *SQL Guide*

Quiescing and unquiescing databases

You can use the **Quiesce** menu option to immediately force all users off a database. Similarly, you can use the **Unquiesce** menu option to return the database to an active state so that all users can access the database.

Prerequisites:

To quiesce or unquiesce a database, you must have SYSADM or DBADM authority.

Procedure:

To quiesce or unquiesce databases using the Control Center:

1. Expand the object tree until you find the database that you want to quiesce or unquiesce.
2. Right-click on the desired database and select **Quiesce** or **Unquiesce** from the pop-up menu. The database will be quiesced or unquiesced immediately.

Related reference:

- “QUIESCE command” in *Command Reference*
- “UNQUIESCE command” in *Command Reference*

Chapter 4. Creating tables and other related table objects

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

Space compression for tables

It might be possible for tables to occupy less space when stored on disk by utilizing such features as compression for data rows, NULL values, and system default values. Through data compression, you might be able to save disk storage space by using fewer database pages to store data. Since more logical data can be stored per page, fewer pages will need to be read in order to access the same amount of logical data. This means that compression can also result in disk I/O savings. I/O speed might also increase because more logical data can be cached in the buffer pool.

To implement data compression in a database system, there are two methods you can employ:

Value compression

This method optimizes space usage for the representation of data, and the storage structures used internally by the database management system (DBMS) to store data. Value compression involves removing duplicate entries for a value, and only storing one copy. The stored copy keeps track of the location of any references to the stored value.

Row compression

This method compresses data rows by replacing repeating patterns that span multiple column values within a row with shorter symbol strings.

Related concepts:

- “Space requirements for database objects” in *Administration Guide: Planning*
- “Data row compression” on page 188
- “Space value compression for existing tables” on page 295
- “Space value compression for new tables” on page 187

Space value 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 concepts:

- “Data row compression” on page 188
- “Space compression for tables” on page 187
- “Space value compression for existing tables” on page 295

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Data row compression

The goal of data row compression is to achieve disk storage space savings. Compression can also lead to disk I/O savings. Also, more data can be cached in the buffer pool, thereby leading to increased bufferpool hit-ratios. There is an associated cost in the form of extra CPU cycles needed to compress and decompress data. The storage savings and performance impact of data row compression are intimately tied to the characteristics of the data within the database, the configuration of the database (layout and tuning) as well as application workload. Only the data on a data page is compressed, as is data in log records. Depending upon UPDATE activity and the positioning of update changes within a data row, there could be an increase in log consumption.

Data row compression is not applicable to Index, Long, LOB and XML data. Row compression and table data replication support are not compatible.

Data row compression uses a static dictionary-based compression algorithm to compress data by row. Compressing data at the row level is advantageous because it allows repeating patterns that span multiple column values within a row to be replaced with shorter symbol strings. In order to compress table data, the table COMPRESS attribute must be set to YES and a compression dictionary must exist for the table. A classic (offline) table reorganization can build a compression dictionary and subsequently compress a table. All the data rows that exist in a table will participate in the building of the compression dictionary. The dictionary will be stored along with the table data rows in the data object portion(s) of the table.

Row compression statistics can be generated using the **RUNSTATS** command and are stored in the system catalog table SYSCAT.TABLES. A compression estimation option is available with the **INSPECT** utility.

Related concepts:

- “Space compression for tables” on page 187
- “Space value compression for existing tables” on page 295
- “Space value compression for new tables” on page 187

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “INSPECT command” in *Command Reference*

- “REORG INDEXES/TABLE command” in *Command Reference*

Table creation

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

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 might 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.

Related concepts:

- “Import Overview” in *Data Movement Utilities Guide and Reference*

Related tasks:

- “Creating and populating a table” on page 217
- “Loading data” in *Data Movement Utilities Guide and Reference*

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Creating a table using the Create Table wizard

Use the Create Table wizard to create new tables in a database. Name the table, determine the columns that you want, and specify the space needed for storing the table data. You can continue by making selections about keys, dimensions, and constraints.

Prerequisites:

To create a table, you must have at least one of the following privileges:

- CREATETAB privilege on the database and USE privilege on the table space, and either:
 - IMPLICIT_SCHEMA authority on the database if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema if the schema name of the table exists
- SYSADM or DBADM authority

Procedure:

To create a table:

1. Open the Create Table wizard: From the Control Center, expand the object tree until you find the Tables folder. Right-click the **Tables** folder and select Create from the pop-up menu. The Create Table wizard opens.
2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is enabled when you specify enough information for the wizard to create a table.

Related tasks:

- “Creating a table in a partitioned database environment” on page 191
- “Creating a table in multiple table spaces” on page 190

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 *SQL Reference, Volume 2*

Creating a table in a partitioned database environment

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

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 *distribution key*. A distribution key is a key that is part of the definition of a table. It determines the database partition on which each row of data is stored.

If you do not specify the distribution key explicitly, the following defaults are used. *Ensure that the default distribution 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 distribution 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 distribution key, the table is created without one (this is allowed only in single-partition database partition groups).

Prerequisites:

Before creating a table that will be physically divided or distributed, you need to consider the following:

- Table spaces can span more than one database partition. The number of database partitions they span depends on the number of database 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 to select an appropriate distribution 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 distribution key. That is, if a distribution key is defined, unique keys and primary keys must include all of the same columns as the distribution key (they might have more columns).

The size limit for one database 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:

To create a table in a partitioned database environment 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>
  DISTRIBUTE BY HASH (<column_name>)
```

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  
DISTRIBUTE BY HASH (MIX_INT)
```

In the preceding example, the table space is MIXTS12 and the distribution key is MIX_INT. If the distribution key is not specified explicitly, it is MIX_CNTL. (If no primary key is specified and no distribution key is defined, the distribution 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 group design” in *Administration Guide: Planning*
- “Database partition groups” in *Administration Guide: Planning*
- “Table collocation” in *Administration Guide: Planning*

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Creating partitioned tables

Partitioned tables use a data organization scheme in which table data is divided across multiple storage objects, called data partitions or ranges, according to values in one or more table partitioning key columns of the table. Data from a given table is partitioned into multiple storage objects based on the specifications provided in the PARTITION BY clause of the CREATE TABLE statement. These storage objects can be in different table spaces, in the same table space, or a combination of both.

You can create a partitioned table by using the Create Table wizard in the DB2 Control Center or by using the CREATE TABLE statement.

Prerequisites:

To create a table, the privileges held by the authorization ID of the statement must include at least one of the following authorities or privileges:

- CREATETAB authority on the database and USE privilege on all the table spaces used by the table, as well as one of:
 - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema
- SYSADM or DBADM authority

Procedure:

You can create a partitioned table from the DB2 Control Center or from the DB2 command line processor (CLP).

To use the Create Table wizard in the DB2 Control Center to create a partitioned table:

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 task.

To use the CLP to create a partitioned table, issue the CREATE TABLE statement:

```
CREATE TABLE <NAME> (<column_name> <data_type> <null_attribute>) IN  
<table space list> PARTITION BY RANGE (<column expression>)  
STARTING FROM <constant> ENDING <constant> EVERY <constant>
```

For example, the following statement creates a table where rows with $a \geq 1$ and $a \leq 20$ are in PART0 (the first data partition), rows with $21 \leq a \leq 40$ are in PART1 (the second data partition), up to $81 \leq a \leq 100$ are in PART4 (the last data partition).

```
CREATE TABLE foo(a INT)  
PARTITION BY RANGE (a) (STARTING FROM (1)  
ENDING AT (100) EVERY (20))
```

Related concepts:

- “Large object behavior in partitioned tables” in *SQL Reference, Volume 1*
- “Table partitioning” in *Administration Guide: Planning*
- “Table partitioning keys” in *Administration Guide: Planning*
- “Understanding clustering index behavior on partitioned tables” in *Performance Guide*
- “Data organization schemes in DB2 and Informix databases” in *Administration Guide: Planning*
- “Understanding index behavior on partitioned tables” in *Performance Guide*
- “Optimization strategies for partitioned tables” in *Performance Guide*
- “Partitioned tables” in *Administration Guide: Planning*
- “Partitioned materialized query table behavior” on page 206

Related tasks:

- “Rotating data in a partitioned table” on page 339
- “Approaches to defining ranges on partitioned tables” on page 195
- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Creating and populating a table” on page 217
- “Approaches to migrating existing tables and views to partitioned tables” on page 198
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “Examples of rolling in and rolling out partitioned table data” on page 342
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338

Details of partitioned tables

This section discusses various approaches to creating partitioned tables.

Approaches to defining ranges on partitioned tables

You can specify a range for each data partition when you create a partitioned table. A partitioned table uses a data organization scheme in which table data is divided across multiple data partitions according to the values of the table partitioning key columns of the table. Data from a given table is partitioned into multiple storage objects based on the specifications provided in the PARTITION BY clause of the CREATE TABLE statement. A range is specified by the STARTING FROM and ENDING AT values of the PARTITION BY clause.

To completely define the range for each data partition, you must specify sufficient boundaries. The following is a list of guidelines to consider when defining ranges on a partitioned table:

- The STARTING clause specifies a low boundary for the data partition range. This clause is mandatory for the lowest data partition range (although you can define the boundary as MINVALUE). The lowest data partition range is the data partition with the lowest specified bound.
- The ENDING (or VALUES) clause specifies a high boundary for the data partition range. This clause is mandatory for the highest data partition range (although you can define the boundary as MAXVALUE). The highest data partition range is the data partition with the highest specified bound.
- If you do not specify an ENDING clause for a data partition, then the next greater data partition must specify a STARTING clause. Likewise, if you do not specify a STARTING clause, then the previous data partition must specify an ENDING clause.
- MINVALUE specifies a value that is smaller than any possible value for the column type being used. MINVALUE and INCLUSIVE or EXCLUSIVE cannot be specified together.
- MAXVALUE specifies a value that is larger than any possible value for the column type being used. MAXVALUE and INCLUSIVE or EXCLUSIVE cannot be specified together.
- INCLUSIVE indicates that all values equal to the specified value are to be included in the data partition containing this boundary.
- EXCLUSIVE indicates that all values equal to the specified value are NOT to be included in the data partition containing this boundary.
- The NULL clause specifies whether null values are to be sorted high or low when considering data partition placement. By default, null values are sorted high. Null values in the table partitioning key columns are treated as positive infinity, and are placed in a range ending at MAXVALUE. If no such data partition is defined, null values are considered to be out-of-range values. Use the NOT NULL constraint if you want to exclude null values from table partitioning key columns. LAST specifies that null values are to appear last in a sorted list of values. FIRST specifies that null values are to appear first in a sorted list of values.
- When using the long form of the syntax, each data partition must have at least one bound specified.

Tip: Before you begin defining data partitions on a table it is important to understand how tables benefit from table partitioning and what factors influence the columns you choose as partitioning columns.

The ranges specified for each data partition can be generated automatically or manually.

Automatically generated:

Automatic generation is a simple method of creating many data partitions quickly and easily. This method is appropriate for equal sized ranges based on dates or numbers.

Examples 1 and 2 demonstrate how to use the CREATE TABLE statement to define and generate automatically the ranges specified for each data partition.

Example 1: Issue a create table statement with the following ranges defined:

```
CREATE TABLE lineitem (  
  l_orderkey    DECIMAL(10,0) NOT NULL,  
  l_quantity    DECIMAL(12,2),  
  l_shipdate    DATE,  
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate))  
  PARTITION BY RANGE(l_shipdate)  
  (STARTING ('1/1/1992') ENDING ('12/31/1992') EVERY 1 MONTH);
```

This statement results in 12 data partitions each with 1 key value (l_shipdate) \geq ('1/1/1992'), (l_shipdate) < ('3/1/1992'), (l_shipdate) < ('4/1/1992'), (l_shipdate) < ('5/1/1992'), ..., (l_shipdate) < ('12/1/1992')(l_shipdate) <= ('12/31/1992').

The starting value of the first data partition is inclusive because the overall starting bound ('1/1/1992') is inclusive (default). Similarly, the ending bound of the last data partition is inclusive because the overall ending bound ('12/31/1992') is inclusive (default). The remaining STARTING values are inclusive and the remaining ENDING values are all exclusive. Each data partition holds n key values where n is given by the EVERY clause. Use the formula (start + every) to find the end of the range for each data partition. The last data partition might have fewer key values if the EVERY value does not divide evenly into the START and END range.

Example 2:

Issue a create table statement with the following ranges defined:

```
CREATE TABLE t(a INT, b INT)  
  PARTITION BY RANGE(b) (STARTING FROM (1) EXCLUSIVE ENDING AT (1000) EVERY (100))
```

This statement results in 10 data partitions each with 100 key values (1 < b <= 101, 101 < b <= 201, ..., 901 < b <= 1000).

The starting value of the first data partition (b > 1 and b <= 101) is exclusive because the overall starting bound (1) is exclusive. Similarly the ending bound of the last data partition (b > 901 b <= 1000) is inclusive because the overall ending bound (1000) is inclusive. The remaining STARTING values are all exclusive and the remaining ENDING values are all inclusive. Each data partition holds n key values where n is given by the EVERY clause. Finally, if both the starting and ending bound of the overall clause are exclusive, the starting value of the first data partition is exclusive because the overall starting bound (1) is exclusive. Similarly the ending bound of the last data partition is exclusive because the overall ending bound (1000) is exclusive. The remaining STARTING values are all exclusive and the ENDING values are all inclusive. Each data partition (except the last) holds n key values where n is given by the EVERY clause.

Manually generated:

Manual generation creates a new data partition for each range listed in the PARTITION BY clause. This form of the syntax allows for greater flexibility when defining ranges thereby increasing your data and LOB placement options. Examples 3 and 4 demonstrate how to use the CREATE TABLE statement to define and generate manually the ranges specified for a data partition.

Example 3:

This statement partitions on two date columns both of which are generated. Notice the use of the automatically generated form of the CREATE TABLE syntax and that only one end of each range is specified. The other end is implied from the adjacent data partition and the use of the INCLUSIVE option:

```
CREATE TABLE sales(invoice_date date, inv_month int NOT NULL
GENERATED ALWAYS AS (month(invoice_date)), inv_year INT NOT
NULL GENERATED ALWAYS AS ( year(invoice_date)), item_id int NOT NULL,
cust_id int NOT NULL) PARTITION BY RANGE (inv_year, inv_month)
(PART Q1_02 STARTING (2002,1) ENDING (2002, 3) INCLUSIVE,
PART Q2_02 ENDING (2002, 6) INCLUSIVE,
PART Q3_02 ENDING (2002, 9) INCLUSIVE,
PART Q4_02 ENDING (2002,12) INCLUSIVE,
PART CURRENT ENDING (MAXVALUE, MAXVALUE));
```

Gaps in the ranges are permitted. The CREATE TABLE syntax supports gaps by allowing you to specify a STARTING value for a range that does not line up against the ENDING value of the previous data partition.

Example 4:

Creates a table with a gap between values 101 and 200.

```
CREATE TABLE foo(a INT)
PARTITION BY RANGE(a)
(STARTING FROM (1) ENDING AT (100),
STARTING FROM (201) ENDING AT (300))
```

Use of the ALTER TABLE statement, which allows data partitions to be added or removed, can also cause gaps in the ranges.

When you insert a row into a partitioned table, it is automatically placed into the proper data partition based on its key value and the range it falls within. If it falls outside of any ranges defined for the table, the insert fails and the following error is returned to the application:

```
SQL0327N The row cannot be inserted into table <tablename>
because it is outside the bounds of the defined data partition ranges.
SQLSTATE=22525
```

Restrictions:

- Table level restrictions:
 - Tables created using the automatically generated form of the syntax (containing the EVERY clause) are constrained to use a numeric or date time type in the table partitioning key.
- Statement level restrictions:
 - MINVALUE and MAXVALUE are not supported in the automatically generated form of the syntax.
 - Ranges are ascending.
 - Only one column can be specified in the automatically generated form of the syntax.

- The increment in the EVERY clause must be greater than zero.
- The ENDING value must be greater than or equal to the STARTING value.

Related concepts:

- “Attributes of detached data partitions” on page 354
- “Data partitions” in *Administration Guide: Planning*
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Creating partitioned tables” on page 193
- “Dropping a data partition” on page 358
- “Approaches to migrating existing tables and views to partitioned tables” on page 198
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Approaches to migrating existing tables and views to partitioned tables

There are three approaches you can use to migrate an existing table or view to a partitioned table:

- When migrating regular tables create a new, empty partitioned table and use the LOAD from CURSOR to move the data from the old table directly into the partitioned table without any intermediate steps.
- When migrating regular tables unload the source table using the export utility or high performance unload, create a new, empty partitioned table and use the LOAD command to populate an empty partitioned table.
- When migrating Union All views create a partitioned table with a single dummy data partition, then attach all of the tables.

Converting regular tables:

To migrate data from a DB2 9.1 table into a partitioned table, use the LOAD command to populate an empty partitioned table.

Example 1:

Suppose you have a regular table t1:

```
CREATE TABLE t1 (c1 int, c2 int);
```

Create a new, empty partitioned table:

```
CREATE TABLE sales_dp (c1 int, c2 int)
PARTITION BY RANGE (c1)
(STARTING FROM 0 ENDING AT 10 EVERY 2);
```

Populate table t1:

```
INSERT INTO t1 VALUES (0,1), (4, 2), (6, 3);
```

To avoid creating a third copy of the data in a flat file, issue the LOAD command to pull the data from an SQL query directly into the new partitioned table.

```
SELECT * FROM t1;
DECLARE c1 CURSOR FOR SELECT * FROM t1;
LOAD FROM c1 OF CURSOR INSERT INTO sales_dp;
SELECT * FROM sales_dp;
```

Drop the old table:

```
DROP TABLE t1;
```

Converting UNION ALL views:

You can convert DB2 9.1 data in a UNION ALL view into a partitioned table. UNION ALL views are used to manage large tables, and achieve easy roll-in and roll-out of table data while providing the performance advantages of branch elimination. Table partitioning accomplishes all of these and is easier to administer. Using the ALTER TABLE ...ATTACH operation, you can achieve conversion with no movement of data in the base table. Indexes and dependent views or materialized query tables (MQT's) must be re-created after the conversion.

The recommended strategy is to create a partitioned table with a single dummy data partition, then attach all of the tables of the union all view. Be sure to drop the dummy data partition early in the process to avoid problems with overlapping ranges.

Example 2:

Create table syntax for first table in the UNION:

```
CREATE TABLE sales_0198(
  sales_date DATE NOT NULL,
  prod_id INTEGER,
  city_id INTEGER,
  channel_id INTEGER,
  revenue DECIMAL(20,2),
  CONSTRAINT ck_date
  CHECK
  (sales_date BETWEEN '01-01-1998' AND '01-31-1998'));
```

Create view syntax for a union all view:

```
CREATE VIEW all_sales AS
(
  SELECT * FROM sales_0198
  WHERE sales_date BETWEEN '01-01-1998' AND '01-31-1998'
  UNION ALL
  SELECT * FROM sales_0298
  WHERE sales_date BETWEEN '02-01-1998' AND '02-28-1998'
  UNION ALL
  ...
  UNION ALL
  SELECT * FROM sales_1200
  WHERE sales_date BETWEEN '12-01-2000' AND '12-31-2000'
);
```

Create a partitioned table with a single dummy partition. The range should be chosen so that it does not overlap with the first data partition to be attached:

```

CREATE TABLE sales_dp (
  sales_date DATE NOT NULL,
  prod_id INTEGER,
  city_id INTEGER,
  channel_id INTEGER,
  revenue DECIMAL(20,2))
PARTITION BY RANGE (sales_date)
(PART dummy STARTING FROM '01-01-1900' ENDING AT '01-01-1900');

```

Attach the first table:

```

ALTER TABLE sales_dp ATTACH PARTITION
STARTING FROM '01-01-1998' ENDING AT '01-31-1998'
FROM sales_0198;

```

Drop the dummy partition:

```

ALTER TABLE sales_dp DETACH PARTITION dummy
INTO dummy;
DROP TABLE dummy;

```

Attach the remaining partitions:

```

ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '02-01-1998' ENDING AT '02-28-1998' FROM sales_0298;
...
ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '12-01-2000' ENDING AT '12-31-2000' FROM sales_1200;

```

Issue the SET INTEGRITY statement to bring the attached data partitions online.

```

SET INTEGRITY FOR sales_dp IMMEDIATE CHECKED
FOR EXCEPTION IN sales_dp USE sales_ex;

```

Create indexes, as appropriate.

Conversion considerations:

Attaching a data partition is allowed unless the value of the SYSCAT.COLUMNS IMPLICITVALUE field in a specific column is a non-null value for both the source column and the target column, and the values do not match. In this case, you must drop the source table and then recreate it.

A column can have a non-null value in the SYSCAT.COLUMNS IMPLICITVALUE field if one of the following conditions are met:

- the column is created as the result of an ALTER TABLE ...ADD COLUMN statement
- the IMPLICITVALUE field is propagated from a source table during attach
- the IMPLICITVALUE field is inherited from a source table during detach
- the IMPLICITVALUE field is set during migration from V8 to V9, where it is determined to be an added column, or might be an added column. If the database is not certain whether the column is added or not, it is treated as added. An added column is a column created as the result of an ALTER TABLE ...ADD COLUMN statement.

To avoid these inconsistencies, it is recommended that you always create the source and target tables involved in an attach operation with the same columns defined. In particular, never use the ALTER TABLE statement to add columns to a target table of an attach operation.

For best practices in avoiding a mismatch when working with partitioned tables, see Resolving a mismatch when trying to attach a data partition to a partitioned table.

Related concepts:

- “Resolving a mismatch when trying to attach a data partition to a partitioned table” on page 348
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Altering a table” on page 297
- “Altering or dropping a view” on page 330

Related reference:

- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “LOAD command” in *Command Reference*
- “SYSCAT.COLUMNS catalog view” in *SQL Reference, Volume 1*

Creating materialized query tables

This section describes various aspects of creating and populating materialized query tables.

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 query 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 queries.

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 might not represent the values of the underlying base table to be used to optimize the processing of the query, the result of the query might *not* accurately represent the data in the underlying table. This might 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 existing data.

A materialized query table defined with REFRESH DEFERRED might 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 9999999999999999, 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 might 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 *SQL Reference, Volume 1*

Related tasks:

- “Refreshing the data in a materialized query table” on page 336
- “Altering materialized query table properties” on page 335
- “Dropping a materialized query or staging table” on page 365

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register” in *SQL Reference, Volume 1*
- “CURRENT REFRESH AGE special register” in *SQL Reference, Volume 1*
- “REFRESH TABLE statement” in *SQL Reference, Volume 2*
- “SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION statement” in *SQL Reference, Volume 2*
- “SET CURRENT REFRESH AGE statement” in *SQL Reference, Volume 2*
- “Restrictions on native XML data store” in *XML Guide*

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 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 205

Related reference:

- “CREATE TABLE statement” in *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 Set Integrity Pending state.

Note: If you want to populate the MQT with SQL insert operations, you need to bring the MQT out of the Set Integrity Pending 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, you will need to bring the MQT out of the Set Integrity Pending 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 *Command Reference*
- “LOAD command” in *Command Reference*

Partitioned materialized query table behavior

All types of materialized query tables (MQTs) are supported with partitioned tables. When working with partitioned MQTs, there are a number of guidelines that can help you to administer attached and detached data partitions most effectively.

The following guidelines and restrictions apply when working with partitioned MQTs or partitioned tables with detached dependents:

- If you issue a DETACH PARTITION operation and there are any dependent tables that need to be incrementally maintained with respect to the detached data partition (these dependents table are referred to as detached dependent tables), then the newly detached table is initially inaccessible. The table will be marked L in the TYPE column of the SYSCAT.TABLES catalog view. This is referred to as a detached table. This prevents the table from being read, modified or dropped until the SET INTEGRITY statement is run to incrementally maintain the detached dependent tables. After the SET INTEGRITY statement is run on all detached dependent tables, the detached table is transitioned to a regular table where it becomes fully accessible.
- To detect that a detached table is not yet accessible, query the SYSCAT.TABDETACHEDDEP catalog view. If any inaccessible detached tables are detected, run the SET INTEGRITY statement with the IMMEDIATE CHECKED option on all the detached dependents to transition the detached table to a regular accessible table. If you try to access a detached table before all its detached dependents are maintained, error code SQL20285N is returned.
- The DATAPARTITIONNUM function cannot be used in a materialized query table (MQT) definition. Attempting to create an MQT using this function returns an error (SQLCODE SQL20058N, SQLSTATE 428EC).
- When creating an index on a table with detached data partitions, the index does not include the data in the detached data partitions unless the detached data partition has a dependent materialized query table (MQT) that needs to be incrementally refreshed with respect to it. In this case, the index includes the data for this detached data partition.
- Altering a table with attached data partitions to an MQT is not allowed.
- Partitioned staging tables are not supported.
- Attaching to an MQT is not directly supported. See Example 1 for details.

- The REFRESH TABLE statement and the SET INTEGRITY statement with the IMMEDIATE CHECKED option is allowed on a partitioned MQT. If an out of range error occur during the refresh, add the missing data partition appropriately and re-execute the REFRESH TABLE statement.

Example 1:

Although the ATTACH operation is not directly supported on partitioned MQTs, you can achieve the same effect by converting a partitioned MQT to an ordinary table, performing the desired roll-in and roll-out of table data, and then converting the table back into an MQT. The following CREATE TABLE and ALTER TABLE statements demonstrate the effect:

```
CREATE TABLE lineitem (
  l_orderkey  DECIMAL(10,0) NOT NULL,
  l_quantity  DECIMAL(12,2),
  l_shipdate  DATE,
  l_year_month INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
  PARTITION BY RANGE(l_shipdate)
  (STARTING ('1/1/1992') ENDING ('12/31/1993') EVERY 1 MONTH);
CREATE TABLE lineitem_ex (
  l_orderkey  DECIMAL(10,0) NOT NULL,
  l_quantity  DECIMAL(12,2),
  l_shipdate  DATE,
  l_year_month INT,
  ts          TIMESTAMP,
  msg        CLOB(32K));

CREATE TABLE quan_by_month (
  q_year_month, q_count) AS
  (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
   FROM lineitem
   GROUP BY l_year_month)
  DATA INITIALLY DEFERRED REFRESH IMMEDIATE
  PARTITION BY RANGE(q_year_month)
  (STARTING (199201) ENDING (199212) EVERY (1),
   STARTING (199301) ENDING (199312) EVERY (1));
CREATE TABLE quan_by_month_ex(
  q_year_month INT,
  q_count      INT NOT NULL,
  ts          TIMESTAMP,
  msg        CLOB(32K));

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
CREATE INDEX qbm ON quan_by_month(q_year_month);

ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;

SET INTEGRITY FOR li_reuse OFF;
ALTER TABLE li_reuse ALTER l_year_month SET GENERATED ALWAYS
AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate));

LOAD FROM part_mqt_rotate.del OF DEL MODIFIED BY GENERATEDIGNORE
MESSAGES load.msg REPLACE INTO li_reuse;

DECLARE load_cursor CURSOR FOR
  SELECT l_year_month, COUNT(*)
  FROM li_reuse
  GROUP BY l_year_month;
LOAD FROM load_cursor OF CURSOR MESSAGES load.msg
REPLACE INTO qm_reuse;

ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
```

```

ENDING '1/31/1994' FROM li_reuse;

SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN lineitem USE lineitem_ex;

ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
ENDING 199401 FROM qm_reuse;

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED
FOR EXCEPTION IN quan_by_month USE quan_by_month_ex;

ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
(SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
FROM lineitem
GROUP BY l_year_month)
DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;

```

Use the SET INTEGRITY statement with the IMMEDIATE CHECKED option to check the attached data partition for integrity violations. This step is required before changing the table back to an MQT. The SET INTEGRITY statement with the IMMEDIATE UNCHECKED option is used to bypass the required full refresh of the MQT. The index on the MQT is necessary to achieve optimal performance. The use of exception tables with the SET INTEGRITY statement is recommended, where appropriate.

Typically, you create a partitioned MQT on a large fact table that is also partitioned. If you do roll out or roll in table data on the large fact table, you must adjust the partitioned MQT manually, as demonstrated in Example 2.

Example 2:

Alter the MQT (quan_by_month) to convert it to an ordinary partitioned table:

```
ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
```

Detach the data to be rolled out from the fact table (lineitem) and the MQT and re-load the staging table li_reuse with the new data to be rolled in:

```
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
LOAD FROM part_mqt_rotate.del OF DEL MESSAGES load.msg REPLACE INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;
```

Prune qm_reuse before doing the insert. This deletes the detached data before inserting the subselect data. This is accomplished with a load replace into the MQT where the data file of the load is the content of the subselect.

```
db2 load from datafile.del of del replace into qm_reuse
```

You can refresh the table manually using INSERT INTO ... (SELECT ...) This is only necessary on the new data, so the statement should be issued before attaching:

```
INSERT INTO qm_reuse
(SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
FROM li_reuse
GROUP BY l_year_month);
```

Now you can roll in the new data for the fact table:

```
ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
ENDING '1/31/1994' FROM TABLE li_reuse;
SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR
EXCEPTION IN li_reuse USE li_reuse_ex;
```

Next, roll in the data for the MQT:

```
ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
ENDING 199401 FROM TABLE qm_reuse;
SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
```

After attaching the data partition, the new data must be verified to ensure that it is in range.

```
ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
(SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
FROM lineitem
GROUP BY l_year_month)
DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

The data is not accessible until it has been validated by the SET INTEGRITY statement. Although the REFRESH TABLE operation is supported, this scenario demonstrates the manual maintenance of a partitioned MQT through the ATTACH PARTITION and DETACH PARTITION operations. The data is marked as validated by the user through the IMMEDIATE UNCHECKED clause of the SET INTEGRITY statement.

Related concepts:

- “Partitioned tables” in *Administration Guide: Planning*
- “Asynchronous index cleanup” in *Performance Guide*
- “Understanding clustering index behavior on partitioned tables” in *Performance Guide*
- “Understanding index behavior on partitioned tables” in *Performance Guide*
- “Optimization strategies for partitioned tables” in *Performance Guide*

Related tasks:

- “Creating a materialized query table” on page 201
- “Creating partitioned tables” on page 193
- “Dropping a materialized query or staging table” on page 365
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352
- “Rotating data in a partitioned table” on page 339
- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Altering a table” on page 297
- “Altering materialized query table properties” on page 335

Related reference:

- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*
- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Creating a new source table using db2look

Creating a new source table might be necessary when the characteristics of the target table do not sufficiently match the characteristics of the source when issuing the ALTER TABLE statement with the ATTACH PARTITION clause. Before creating a new source table, you can attempt to correct the mismatch between the existing source table and the target table.

If attempts to correct the mismatch fail, error SQL20408N or SQL20307N is returned.

Prerequisites:

To create a table, the privileges held by the authorization ID of the statement must include at least one of the following authorities and privileges:

- CREATETAB authority on the database and USE privilege on the table space, as well as one of:
 - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema
- SYSADM or DBADM authority

Procedure:

To create a new source table:

1. Use the db2look command to produce the CREATE TABLE statement to create a table identical to the target table:

```
db2look -d <source database name> -t <target database name> -e -p
```

2. Remove the partitioning clause from the **db2look** output and change the name of the table created to a new name (in this example, referred to here as sourceC).
3. Next, load all of the data from the original source table to the newly created source table, sourceC using a LOAD FROM CURSOR command:

```
DECLARE mycurs CURSOR FOR SELECT * FROM source
LOAD FROM mycurs OF CURSOR REPLACE INTO sourceC
```

If this command fails because the original data is incompatible with the definition of table sourceC, you must transform the data in the original table as it is being transferred to sourceC.

4. After the data has been successfully copied to sourceC, submit the ALTER TABLE target ...ATTACH sourceC statement.

Related concepts:

- “Resolving a mismatch when trying to attach a data partition to a partitioned table” on page 348
- “Partitioned tables” in *Administration Guide: Planning*
- “Tables” in *SQL Reference, Volume 1*
- “Mimicking databases using db2look” in *Troubleshooting Guide*

Related tasks:

- “Altering a table” on page 297

- “Altering partitioned tables” on page 336
- “Attaching a data partition” on page 346

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “db2look - DB2 statistics and DDL extraction tool command” in *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 materialized query table, for which the staging table is created, 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.
4. Partitioned hierarchy tables and partitioned typed tables are not supported. (Partitioned tables are tables where data is partitioned into multiple storage objects based on the specifications provided in the PARTITION BY clause of the CREATE TABLE statement.)

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 might 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 might 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:

- “Altering materialized query table properties” on page 335
- “Creating a materialized query table” on page 201
- “Dropping a materialized query or staging table” on page 365
- “Refreshing the data in a materialized query table” on page 336

Related reference:

- “SET INTEGRITY statement” in *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.

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.

Prerequisites:

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

Restrictions:

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:

To create a user-defined temporary table, use the DECLARE GLOBAL TEMPORARY TABLE statement. The statement is used from within an application.

Example of how to define a temporary table:

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
  LIKE emp1tab1
  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 emp1tab1. The implicit definition only includes the column name, data type, 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.

If 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 159

Related reference:

- “DECLARE GLOBAL TEMPORARY TABLE statement” in *SQL Reference, Volume 2*
- “ROLLBACK statement” in *SQL Reference, Volume 2*
- “SAVEPOINT statement” in *SQL Reference, Volume 2*

Creating range-clustered tables

This section provides examples and guidelines for range-clustered tables, including how the compiler works with these types of tables.

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, the following information is included:

- School ID
- Program ID

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

In this case, the student records are based solely 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 1 million student records are allowed, 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, the student records are 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 might 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, 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 *SQL Reference, Volume 2*

How the query 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:

- “Guidelines for using range-clustered tables” on page 216
- “Range-clustered tables” in *Administration Guide: Planning*

Guidelines for using range-clustered tables

When working with the DB2 database manager 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 *Administration Guide: Planning*
- “Examples of range-clustered tables” on page 213

Creating typed tables

This section describes how to create and populate typed tables.

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.

Restrictions:

Partitioned hierarchy tables and partitioned typed tables are not supported. (Partitioned tables are tables where data is partitioned into multiple storage objects based on the specifications provided in the `PARTITION BY` clause of the `CREATE TABLE` statement.)

Procedure:

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

Related concepts:

- “Typed tables” in *SQL Guide*

Related tasks:

- “Populating a typed table” on page 217
- “Creating a structured type hierarchy” in *SQL Guide*
- “Creating typed tables” in *SQL Guide*
- “Creating typed views” in *SQL Guide*
- “Dropping typed tables” in *SQL Guide*

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TYPE (Structured) statement” in *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 *SQL Guide*
- “Typed tables” in *SQL Guide*

Related tasks:

- “Creating a hierarchy table or a typed table” on page 216
- “Creating typed tables” in *SQL Guide*
- “Dropping typed tables” in *SQL Guide*
- “Storing objects in typed table rows” in *SQL Guide*

Related reference:

- “CREATE TYPE (Structured) statement” in *SQL Reference, Volume 2*

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 a 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.

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,
  FIRSTNAME 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*.

Refer to the topics in the related information sections for other options that you should consider when creating and populating a table.

Related concepts:

- “Import Overview” in *Data Movement Utilities Guide and Reference*
- “Load overview” in *Data Movement Utilities Guide and Reference*
- “Moving data across platforms - file format considerations” in *Data Movement Utilities Guide and Reference*
- “Comparing IDENTITY columns and sequences” on page 235
- “Large object (LOB) column considerations” on page 221
- “Table creation” on page 189

- “User-defined types (UDTs)” on page 246

Related tasks:

- “Creating a hierarchy table or a typed table” on page 216
- “Creating a materialized query table” on page 201
- “Creating a sequence” on page 234
- “Creating a table in a partitioned database environment” on page 191
- “Creating a table in multiple table spaces” on page 190
- “Creating a user-defined temporary table” on page 212
- “Defining a generated column on a new table” on page 219
- “Defining an identity column on a new table” on page 220
- “Defining dimensions on a table” on page 235
- “Defining a unique constraint on a table” on page 223

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “DECLARE GLOBAL TEMPORARY TABLE statement” in *SQL Reference, Volume 2*
- “INSERT statement” in *SQL Reference, Volume 2*
- “IMPORT Command” in *Command Reference*
- “LOAD command” in *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.

Defining columns

This section discusses how to define generated and identify columns on a new table.

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 321

Related reference:

- “CREATE INDEX statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “SELECT statement” in *SQL Reference, Volume 2*
- “Restrictions on native XML data store” in *XML Guide*

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 in which 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.

Other 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 the DB2 database manager: ALWAYS or BY DEFAULT.

An identity column defined as GENERATED ALWAYS is given values that are always generated by the DB2 database manager. 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.

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 might 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:

To define an identity column on a new table, use the AS IDENTITY clause on the CREATE TABLE statement.

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.

Related concepts:

- “Comparing IDENTITY columns and sequences” on page 235

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Large object (LOB) column considerations

Before creating a table that contains large object columns, you need to answer the following questions:

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. For example:

```

CREATE TABLE EMPLOYEE
  (EMPNO      CHAR(6)      NOT NULL PRIMARY KEY,
   FIRSTNAME  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 might 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 in which 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,
   FIRSTNAME  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).

Note: When moving LOBs, small LOBs are stored in the applications heap and large LOBs are stored in temporary tables within a 4 KB page size temporary table space.

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 Version 8 and later: System catalogs that use LOB columns and might take up more space than in previous versions.

3. Do you want better performance for LOB columns, including those LOB columns in the 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 *Administration Guide: Planning*

Related reference:

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

Defining keys and constraints

This section discusses how to define constraints and sequences on tables. 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 on a table

Unique constraints ensure that every value in the specified key is unique. A table can have any number of unique constraints, with one unique constraint defined as a primary key.

Restrictions:

A unique constraint might 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:

- “Constraints” in *SQL Reference, Volume 1*
- “Keys” in *SQL Reference, Volume 1*

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Defining referential constraints on tables

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 keys in a referential constraint” on page 226
- “REFERENCES clause in a referential constraint” on page 227

Related tasks:

- “Adding foreign keys” on page 310

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE SCHEMA statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Foreign keys in a referential constraint

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. A foreign key makes its table dependent on another table called a parent table. The values in the column or set of columns that make up the foreign key in one table must match the unique key or primary key values of the other table.

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.

When working with foreign keys you can:

- Create a table with zero or more foreign keys.
- Define foreign keys when a table is created or altered.
- Drop foreign keys when a table is altered.

Related tasks:

- “Defining a unique constraint on a table” on page 223
- “Defining referential constraints on tables” on page 224

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

REFERENCES clause in a referential constraint

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 this 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 keys in a referential constraint” on page 226

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Table constraint implications for utility operations

If the table being loaded into has referential integrity constraints, the load utility places the table into the Set Integrity Pending state to inform you that the SET INTEGRITY statement is required to be run on the table, in order to verify the referential integrity of the loaded rows. After the load utility has completed, you will need to issue the SET INTEGRITY statement to carry out the referential integrity checking on the loaded rows and to bring the table out of the Set Integrity Pending state. For example, if the DEPARTMENT and EMPLOYEE tables are the only tables that have been placed in Set Integrity Pending state, you can execute the following statement:

```
SET INTEGRITY FOR DEPARTMENT ALLOW WRITE ACCESS, EMPLOYEE ALLOW WRITE ACCESS
IMMEDIATE CHECKED FOR EXCEPTION IN DEPARTMENT USE DEPARTMENT_EX,
IN EMPLOYEE USE EMPLOYEE_EX
```

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:

- “Checking for integrity violations following a load operation” in *Data Movement Utilities Guide and Reference*
- “Import Overview” in *Data Movement Utilities Guide and Reference*
- “Load overview” in *Data Movement Utilities Guide and Reference*

Related reference:

- “SET INTEGRITY statement” in *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 *SQL Reference, Volume 1*


Related tasks:

- “Adding a table check constraint” on page 314
- “Checking for constraint violations using SET INTEGRITY” on page 230
- “Making a table in no data movement mode fully accessible” on page 238

Related reference:

- “ALTER SERVER statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Defining distribution keys

 During table alteration, distribution keys can only be defined if the table resides in a single-partition database partition group.

Procedure:

To define distribution keys using the Control Center:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the Keys page, click **Add Partitioning**. The Define Partitioning Key window opens.
3. Select the columns that you want to add as distribution key columns and move them to the **Selected columns** box.

Related concepts:

- “Table partitioning keys” in *Administration Guide: Planning*

Adding check constraints

You can add check constraints to your table or nickname. A check constraint sets restrictions on data added to the table or nickname. Check constraints are enforced when rows in the table or nickname are inserted or updated. You can define a check constraint that references a single column.

Prerequisites:

To add check constraints, you must have at least one of the following privileges on the table to be altered:

- ALTER privilege
- CONTROL privilege
- SYSADM or DBADM authority
- ALTERIN privilege on the schema of the table

Procedure:

To add check constraints using the Control Center:

1. Open the Alter Table notebook if you are adding a unique key to a table: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want in the contents pane and select **Alter** from the pop-up menu. The Alter Table notebook opens.
If you are adding check constraints on a nickname, open the Alter Nickname notebook.
2. On the Check Constraints page, click **Add**. The Add Check Constraint window opens.
3. For as many check constraints as you are adding: Specify the check condition for the constraint that you are defining, type a name for the check constraint, and optionally type a comment to document the new check constraint.

To add a check constraint using the command line, use the **ALTER TABLE** statement.

Related tasks:

- “Adding a table check constraint” on page 314
- “Changing check constraints” on page 314
- “Defining a table check constraint” on page 228
- “Dropping a table check constraint” on page 317

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Checking for constraint violations using SET INTEGRITY

Typically, you need to manually perform integrity processing for a table in three situations:

- After loading data into a table
- When altering a table by adding constraints on the table
- When altering a table to add a generated column

The load operation causes a table to be put into Set Integrity Pending state automatically if the table has constraints defined on it or if it has dependent foreign key tables, dependent materialized query tables, or dependent staging tables. When the load operation is completed, you can verify the integrity of the loaded data and you can turn on constraint checking for the table. If the table has dependent foreign key tables, dependent materialized query tables, or dependent staging tables, they will be automatically put into Set Integrity Pending state. You will need to use the Set Integrity window to perform separate integrity processing on each of these tables.

If you are altering a table by adding a foreign key, a check constraint or a generated column, you need to turn off constraint checking before you alter the table. After you add the constraint, you need to check the existing data for violations to the newly added constraint and you need to turn constraint checking back on. In addition, if you are loading data into the table, you cannot activate constraint checking on the table until you complete loading data into it. If you are importing data into the table, you should activate constraint checking on the table before you import data into it.

Constraints checking refers to checking for constraints violations, foreign key violations, and generated columns violations. Integrity processing refers to populating identity and generated columns, refreshing materialized query tables, and propagating to staging tables, in addition to performing constraints checking.

Normally, referential integrity and check constraints on a table are automatically enforced, materialized query tables are automatically refreshed immediately, and staging tables are automatically propagated. In some situations, you might need to manually change this behavior.

Prerequisites:

- To turn on constraint checking for a table and performing integrity processing on the table, you need one of the following:
 - SYSADM or DBADM authority
 - CONTROL privileges on the tables being checked, and if exceptions are being posted to one or more tables, INSERT privilege on the exception tables

- CONTROL privilege on all descendent foreign key tables, descendent immediate materialized query tables, and descendent immediate staging tables that will implicitly be placed in the Set Integrity Pending state by the statement
- LOAD authority, and if exceptions are being posted to one or more tables:
 - SELECT and DELETE privilege on each table being checked
 - INSERT privilege on the exception tables
- To turn on constraint checking for a table without performing integrity processing on the table, you need one of the following:
 - SYSADM or DBADM authority
 - CONTROL privileges on the tables being checked
 - CONTROL privilege on each descendent foreign key table, descendent immediate materialized query table, and descendent immediate staging table that will implicitly be placed in the Set Integrity Pending state by the statement
- To turn off constraint checking, immediate refreshing, or immediate propagation for tables, you need one of the following:
 - SYSADM or DBADM authority
 - CONTROL privilege on the table, and on all descendent foreign key tables, descendent immediate materialized query tables, and descendent immediate staging tables that will have their integrity checking turned off by the statement
 - LOAD authority

Procedure:

To check for constraint violations using the Control Center:

1. Open the Set Integrity window: From the Control Center, expand the object tree until you find the **Tables** folder. Click on the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Set Integrity** from the pop-up menu. The Set Integrity window opens.
2. Review the Current Integrity Status of the table you are working with.
3. To turn on constraint checking for a table and not check the table data:
 - a. Select the **Immediate and unchecked** radio button.
 - b. Specify the type of integrity processing that you are turning on.
 - c. Select the **Full Access** radio button to immediately perform data movement operations against the table (such as reorganize or redistribute). However, note that subsequent refreshes of dependent materialized query tables will take longer. If the table has an associated materialized query table, it is recommended that you do not select this radio button in order to reduce the time needed to refresh the materialized query table.
4. To turn on constraint checking for a table and check the existing table data:
 - a. Select the **Immediate and checked** radio button.
 - b. Select which type of integrity processing that you want to perform. If the **Current integrity status** shows that the constraints checked value for the materialized query table is incomplete, you cannot incrementally refresh the materialized query table.
 - c. Optional: If you want identity or generated columns to be populated during integrity processing, select the **Force generated** check box.
 - d. If the table is not a staging table, make sure that the **Prune** check box is unchecked.
 - e. Select the **Full Access** radio button to immediately perform data movement operations against the table.
 - f. Optional: Specify an exception table. Any row that is in violation of a referential or check constraint will be deleted from your table and copied to the exception table. If you do not specify an exception table, when a constraint is violated, only the first violation detected is returned to you and the table is left in the Set Integrity Pending state.
5. To turn off constraint checking, immediate refreshing, or immediate propagation for a table:
 - a. Select the **Off** radio button. The table will be put in Set Integrity Pending state.
 - b. Use the **Cascade** option to specify whether you want to cascade immediately or defer cascading. If you are cascading immediately, use the **Materialized Query Tables**, **Foreign Key Tables**, and **Staging Tables** check boxes to indicate the tables to which you want to cascade.

Note: If you turn off constraint checking for a parent table and specify that you want to cascade the changes to foreign key tables, the foreign key constraints of all of its descendent foreign key tables are also turned off. If you turn off constraint checking for a underlying table and specify that you want to cascade the check pending state to materialized query tables, the refresh immediate properties of all its dependent materialized query tables are also turned off. If you turn off constraint checking for a underlying table and specify that you want to cascade the Set Integrity Pending state to staging tables the propagate immediate properties of all its dependent staging tables are also turned off.

To check for constraint violations using the command line, use the SET INTEGRITY statement.

Troubleshooting tip:

Symptom

You receive the following error message when you try to turn on constraints checking, immediate refresh, or immediate propagation for a table:

DB2 Message

Cannot check a dependent table TABLE1 using the SET INTEGRITY statement while the parent table or underlying table TABLE2 is in the Set Integrity Pending state or if it will be put into the Set Integrity Pending state by the SET INTEGRITY statement.

Where TABLE1 is the table for which you are trying to turn on constraints checking, immediate refresh, or immediate propagation and it is dependent on TABLE2.

Possible cause

Constraint checking, immediate refresh, or immediate propagation cannot be turned on for a table that has a parent or underlying table in Set Integrity Pending.

Action

Bring the parent or underlying table out of Set Integrity Pending by turning on constraint checking for the table. Begin with the table identified as the parent or underlying table in the DB2 message. If that table is dependent on another table, you need to turn on constraint checking in a top-down approach from the table at the top of the dependency chain.

Attention: If the selected table has a cyclical referential constraint relationship with one or more tables, you cannot use the Set Integrity window to turn on constraint checking. In this case, you must use the Command Editor to issue the SQL SET INTEGRITY command.

Related tasks:

- “Adding check constraints” on page 229
- “Changing check constraints” on page 314

Related reference:

- “C samples” in *Samples Topics*
- “Command Line Processor (CLP) samples” in *Samples Topics*
- “JDBC samples” in *Samples Topics*
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*
- “SQLJ samples” in *Samples Topics*

Defining an informational constraint

An *informational constraint* is a rule that can be used by the SQL and XQuery compiler but is not enforced by the database manager. The query compiler includes a rewrite query stage which transforms SQL and XQuery 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 *SQL Reference, Volume 1*
- “Query rewriting methods and examples” in *Performance Guide*
- “The SQL and XQuery compiler process” in *Performance Guide*

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *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 point in time before the database failure, 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.

Related concepts:

- “Comparing IDENTITY columns and sequences” on page 235
- “Sequences” on page 461

Related reference:

- “CREATE SEQUENCE statement” in *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 might 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 or XQuery 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 and XQuery 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:

- “Creating a sequence” on page 234
- “Defining a generated column on a new table” on page 219
- “Defining a generated column on an existing table” on page 321

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 multidimensional 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:

To define dimensions using the Control Center:

1. Open the Create Table wizard: From the Control Center, expand the object tree until you see the **Tables** folder. Right-click the **Tables** folder and select **Create** from the pop-up menu. The Create Table wizard opens.
2. On the Dimensions page, click **Add**. The Dimension window opens.
3. In the **Available columns** box, select the columns that you want in the column group of the dimension and click the > push button to move the column or columns to the **Selected columns** box.
4. Click **Apply** to add a dimension to the **Dimensions** list on the Dimension page.

To define dimensions using the command line, specify each dimension 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 might 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 might 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 *Administration Guide: Planning*

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Loading data into a table using the Load wizard

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

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

Note: If the table you are working with is a replication source, the changes made will not be captured in replication.

Prerequisites:

To load data into a table, you must have one of the following authorities:

- SYSADM authority
- DBADM authority
- LOAD authority on the database and:
 - INSERT privilege on the table if you load data in INSERT mode, TERMINATE mode (to terminate a previous load operation), or RESTART mode (to restart a previous load insert operation)
 - INSERT and DELETE privilege on the table if you load data in REPLACE mode, TERMINATE mode (to terminate a previous load replace operation), or RESTART mode (to restart a previous load replace operation)

- INSERT privilege on the exception table, if one is used during the load operation

Note: Because all load processes (and all DB2 server processes in general) are owned by the instance owner, and all these processes use the identification of the instance owner to access the required files, the instance owner must have read access to input data files. The input data files must be readable by the instance owner, regardless of who performs the load operation.

Procedure:

To load data into a table:

1. Open the Load wizard: From the Control Center, expand the object tree until you find the Tables folder. Click on the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window (the contents pane). Right-click the table you want in the contents pane and select **Load** from the pop-up menu. The Load wizard opens.
2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is enabled when you specify enough information for the wizard to create the load task.

Related concepts:

- “LOAD authority” on page 511
- “Load considerations for MDC tables” in *Administration Guide: Planning*

Related tasks:

- “Enabling parallelism for loading data” on page 10

Related reference:

- “LOAD command” in *Command Reference*

Making a table in no data movement mode fully accessible

Use the Set Integrity window to make a table in no data movement mode fully accessible.

Note: Use these steps only if the table you are working with is in no data movement mode. If the table you are working with is in Set Integrity Pending state, you must turn on constraint checking. See Checking for constraint violations using SET INTEGRITY.

Prerequisites:

To bring a table from no data movement mode to full access mode, you need the following authorities:

- SYSADM or DBADM authority
- CONTROL privilege on the tables that are moving from no data movement to full access

Procedure:

To make a table in no data movement mode fully accessible using the Control Center:

1. Open the Set Integrity window: From the Control Center, expand the object tree until you find the **Tables** folder. Click on the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Set Integrity** from the pop-up menu. The Set Integrity window opens.
2. Review the **Current integrity status** of the table you are working with.
3. Select the **Full Access** radio button.

To make a table in no data movement mode fully accessible using the command line, use the **SET INTEGRITY** statement.

Related concepts:

- “Constraints” in *SQL Reference, Volume 1*

Related tasks:

- “Adding a table check constraint” on page 314
- “Checking for constraint violations using SET INTEGRITY” on page 230
- “Defining a table check constraint” on page 228

Related reference:

- “SET INTEGRITY statement” in *SQL Reference, Volume 2*

Quiescing tables

You can change the quiesce mode of a table and its table spaces. When you quiesce a table and its table spaces, locks are placed on the table and table spaces. The type of lock depends on the quiesce mode.

Prerequisites:

To change the quiesce mode of a table, you must have one of the following authorities: SYSADM, SYSCtrl, SYSMAINT, DBADM, or LOAD.

Procedure:

To change the quiesce mode for a table using the Control Center:

1. Open the Quiesce window: From the Control Center, expand the object tree until you find the **Tables** folder. Click on the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Quiesce** from the pop-up menu. The Quiesce window opens.
2. If you are turning on the quiesce mode or updating the quiesce mode to a higher mode:
 - a. Make sure that the **Quiesce** radio button is selected.
 - b. Select one of the following three modes:
 - Shared** Puts the table in shared mode. In this mode, all users (yourself included) can read but not change the table data.
 - Intent to update**
Puts the table in update mode. In this mode, only you can update the table data. Other users can read, but not update the data.
 - Exclusive**
Puts the table in exclusive mode. In this mode, only you can read or update the table data.

If the table is already in one quiesce mode, you can change it to a higher (more exclusive) mode. For example, if the table is already in shared mode, you can change it to intent to update, or to exclusive mode.

However, you cannot change a higher mode to a lower mode. Exclusive is higher than intent to update, which is higher than shared.
3. If you are resetting a table's quiesce mode, select the **Quiesce reset** radio button.

To change the quiesce mode for a table using the command line, use the **QUIESCE** command.

Related reference:

- "QUIESCE command" in *Command Reference*
- "QUIESCE TABLESPACES FOR TABLE command" in *Command Reference*

Defining triggers

This section discusses trigger creation and dependencies.

Creating triggers

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 might 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, 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 242
- “Updating view contents using triggers” on page 328
- “INSERT, UPDATE, and DELETE triggers” in *SQL Guide*
- “Trigger creation guidelines” in *SQL Guide*
- “Triggers in application development” in *SQL Guide*

Related tasks:

- “Dropping a trigger” on page 329
- “Defining actions using triggers” in *SQL Guide*
- “Defining business rules using triggers” in *SQL Guide*

Related reference:

- “CREATE TRIGGER statement” in *SQL Reference, Volume 2*
- “Restrictions on native XML data store” in *XML Guide*

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:

- “Updating view contents using triggers” on page 328

Related tasks:

- “Creating triggers” on page 240
- “Dropping a trigger” on page 329

Related reference:

- “CREATE TRIGGER statement” in *SQL Reference, Volume 2*
- “DROP statement” in *SQL Reference, Volume 2*

Defining UDFs and UDTs

This section discusses user-defined functions (UDFs) or methods.

User-defined functions (UDFs) or methods

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 the DB2 database manager. An example of a column function is the built-in function AVG(). An external column UDF cannot be defined to the DB2 database manager, 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 might 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:

- “General rules for updating catalog statistics manually” in *Performance Guide*
- “Statistics for user-defined functions” in *Performance Guide*

- “Scalar functions” in *SQL Reference, Volume 1*
- “Table functions” in *SQL Reference, Volume 1*
- “User-defined functions” in *SQL Reference, Volume 1*
- “DB2 user-defined functions and methods” in *SQL Guide*

Related tasks:

- “Creating a function mapping in a federated database” on page 244
- “Creating a function template in a federated system” on page 245

Related reference:

- “Functions” in *SQL Reference, Volume 1*
- “CREATE FUNCTION statement” in *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

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.

Prerequisites:

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.

Restrictions:

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.

Procedure:

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')
```

Related concepts:

- “Host language program mappings with transform functions” in *SQL Guide*

Related tasks:

- “Creating a function template in a federated system” on page 245

Related reference:

- “CREATE FUNCTION MAPPING statement” in *SQL Reference, Volume 2*

Creating a function template in a federated system

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 within the DB2 database manager.

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 tasks:

- “Creating a function mapping in a federated database” on page 244

Related reference:

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

User-defined types (UDTs)

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:

- “User-defined structured types” on page 248

Related tasks:

- “Creating a user-defined distinct type” on page 247

Related reference:

- “Data types” in *SQL Reference, Volume 1*
- “User-defined types” in *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:

Use the **CREATE DISTINCT TYPE** statement to create a distinct type:

```
CREATE DISTINCT TYPE <distinct_type_name> AS <value> WITH COMPARISONS
```

For example, 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,
   FIRSTNAME  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.

Related concepts:

- “User-defined types (UDTs)” on page 246

Related reference:

- “CREATE DISTINCT TYPE statement” in *SQL Reference, Volume 2*
- “CREATE FUNCTION (Sourced or Template) statement” in *SQL Reference, Volume 2*
- “Data types” in *SQL Reference, Volume 1*

- “User-defined types” in *SQL Reference, Volume 1*

User-defined structured types

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:

- “Structured type hierarchies” in *SQL Guide*
- “User-defined structured types” in *SQL Guide*

Related tasks:

- “Creating a structured type hierarchy” in *SQL Guide*
- “Creating structured types” in *SQL Guide*

Related reference:

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

Creating a type mapping in a federated system

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 *SQL Reference, Volume 2*
- “Data type mappings between DB2 and OLE DB” in *Developing ADO.NET and OLE DB Applications*

Source data types

The following data types are available as sources for a distinct type. Each one appears with a short description of how they are defined.

Integer

Holds an integer value between $-2\,147\,483\,648$ and $2\,147\,482\,647$ and uses four bytes of database storage. Integers have a precision of 10 digits.

Smallint

Hold an integer value between -32768 and 32767 and uses two bytes of database storage. Smallints have a precision of 5 digits.

Bigint

Holds an integer value between $-9\,223\,372\,036\,854\,775\,808$ and $9\,223\,372\,036\,854\,775\,807$ and uses 8 bytes of storage space. Large number processing is often more efficient with bigint than with decimal types.

Real

A floating-point number. Real uses 4 bytes of storage space. Single Precision and Float (where float has a length between 1 and 24) are synonyms for Real.

Double

A floating-point number. Double uses 8 bytes of storage space. Double Precision and Float (where float has a length between 25 and 53) are synonyms for Double.

Decimal

A number in decimal form. The precision of a decimal type is the total number of digits. The scale of a decimal is the number of digits in the fractional part. Decimal data is stored in packed format. If a decimal data type is to be used in a C program, the host variable must be declared as a double.

Character

A character string of fixed length. It can be between 1 and 254 characters. Every entry will take up the same amount of space in the database, regardless of whether or not the entered string is as long as the set length.

Varchar

A variable length character string with a maximum length that can be set to no more than 4000 characters. The database will use only the necessary amount of storage space to store the value. Changing the values at a later time can eventually result in performance degradation.

Long Varchar

A variable length character string with a maximum length that can be set to no more than 32 700 characters.

BLOB

A binary large object string.

CLOB

A character large object string.

DBCLOB

A variable-length graphic string with a maximum length of 1 073 741 823 double-byte characters.

Graphic

A fixed-length graphic string of double-byte characters.

Vargraphic

A variable-length graphic string with a maximum length of 2000 double-byte characters.

Long Vargraphic

A variable-length graphic string with a maximum length of 16 350 double-byte characters.

Date Stores a date value. It is stored internally as a packed string of 4 bytes, and appears externally as a string with a length of 10 bytes.

Time Stores a time value. It is stored internally as a packed strong of 3 bytes, and appears externally as a string with length of 8 bytes.

Timestamp

Stores a timestamp value. It is stored internally as a packed string of 10 bytes and appears externally as a string with a length of 26 bytes.

Related tasks:

- “Creating a user-defined distinct type” on page 247

Related reference:

- “Length limits for source data types” on page 250

Length limits for source data types

Certain source data types require a value to be entered in the **Length** field. For each data type, the length has different limits, and different meanings as explained here:

Character

Length of the fixed-length character string, which can range from 1 to 254.

Varchar

Maximum length of the varying-length character string, which can range from 1 to 4000.

BLOB

Maximum length of the BLOB string, which can range from 1 to 2147483647. If a LOB unit of Kbytes, Mbytes, or Gbytes is specified in the **LOB unit** box, the maximum value of **Length** is limited as follows:

Kbytes

The maximum value for this field is 2097152. Each Kbyte is equivalent to 1024 bytes, so the maximum length in bytes is 1024 times the integer value you specify in this field.

Mbytes

The maximum value for this field is 2048. Each Mbyte is equivalent to 1048576 bytes, so the maximum length in bytes is 1048576 times the integer value you specify in this field.

Gbytes

The maximum value for this field is 2. Each GByte is equivalent to 1073741824 bytes, so the maximum length in bytes is 1073741824 times the integer value you specify in this field.

CLOB

Maximum length of the CLOB string, which can range from 1 to 2147483647. If a LOB unit of Kbytes, Mbytes, or Gbytes is specified in the **LOB unit** box, the maximum value of **Length** is limited as follows:

Kbytes

The maximum value for this field is 2097152. Each Kbyte is equivalent to 1024 bytes, so the maximum length in bytes is 1024 times the integer value you specify in this field.

Mbytes

The maximum value for this field is 2048. Each Mbyte is equivalent to 1048576 bytes, so the maximum length in bytes is 1048576 times the integer value you specify in this field.

Gbytes

The maximum value for this field is 2. Each Gbyte is equivalent to 1073741824 bytes, so the maximum length in bytes is 1073741824 times the integer value you specify in this field.

DCLOB

Maximum length of the DCLOB string, which can range from 1 to 1073741823 double-byte characters. If a LOB unit of Kbytes, Mbytes, or Gbytes is specified in the **LOB unit** box, the maximum value of **Length** is limited as follows:

Kbytes

The maximum value for this field is 1048576. Each Kbyte is equivalent to 1024 bytes, so the maximum length in double bytes is 1024 times the integer value you specify in this field.

Empties

The maximum value for this field is 1024. Each Mbyte is equivalent to 1048576 bytes, so the maximum length in double bytes is 1048576 times the integer value you specify in this field.

Gbytes

The maximum value for this field is 1. Each Gbyte is equivalent to 1073741824 bytes, so the maximum length in double bytes is 1073741824 times the integer value you specify in this field.

Graphic

Length of the fixed-length graphic string, which can range from 1 to 127. If the length is not specified, a length of 1 is assumed.

Vargraphic

Maximum length of the varying-length graphic string, which can range from 1 to 2000.

Related tasks:

- “Creating a user-defined distinct type” on page 247

Related reference:

- “CREATE DISTINCT TYPE statement” in *SQL Reference, Volume 2*
- “Source data types” on page 249

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.

A typed view is based on a predefined structured type. You can create a typed view using the CREATE VIEW statement.

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 might 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 might 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 *SQL Reference, Volume 1*
- “Controlling access to data with views” on page 525
- “Table and view privileges” on page 515
- “Updating view contents using triggers” on page 328

Related tasks:

- “Altering or dropping a view” on page 330
- “Recovering inoperative views” on page 331
- “Removing rows from a table or view” on page 307
- “Creating typed views” in *SQL Guide*

Related reference:

- “CREATE VIEW statement” in *SQL Reference, Volume 2*
- “INSERT statement” in *SQL Reference, Volume 2*

Creating an alias

An alias is an indirect method of referencing a table, nickname, or view, so that an SQL or XQuery 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 or XQuery 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 the SQL or XQuery 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 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 *SQL Reference, Volume 1*

Related reference:

- “CREATE ALIAS statement” in *SQL Reference, Volume 2*

Creating indexes

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 a multi-dimensional clustering (MDC) 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. For a partitioned MDC table, the MDC block indexes are stored in the default table space or the table space defined in the INDEXES IN clause of CREATE TABLE statement.

In partitioned database environments, where table data is distributed across database partitions, the index object is distributed across the database partitions in the same manner as the table. For regular tables this would mean one index object per database partition. When creating partitioned tables, non-partitioned indexes are created. Each index is an independent object shared among all the data partitions of the table and stored in a single table space.

Table 18. Index types and locations.

Table type	Index description	Index storage location
MDC	Block	By default, the index is stored in same table space as the table data, however, you can specify different table space using the INDEX IN clause on the CREATE TABLE statement.
Distributed	Independent (not shared)	The index is distributed across the database partitions.
Partitioned	Non-partitioned (shared)	The index is stored in a single table space.

Table 18. Index types and locations. (continued)

Table type	Index description	Index storage location
Partitioned	Partitioned (shared)	The index is stored in a single table space in a separate index object. You can also specify a different table space for each index on the table.

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, or for partitioned tables, the index location can be overridden using the IN clause of the CREATE INDEX statement. This allows different table spaces to be specified for different indexes, as required.

For example, to create an index on a partitioned table, assume there is already a partitioned table foo (a int, b int, c int). To create a unique index, a_idx in the table space my_tbsp, use this command:

```
CREATE UNIQUE INDEX a_idx ON foo ( a ) IN my_tbsp
```

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 *Data Movement Utilities Guide and Reference*
- “Understanding index behavior on partitioned tables” in *Performance Guide*
- “Index cleanup and maintenance” in *Performance Guide*
- “Relational index performance tips” in *Performance Guide*
- “Relational index planning tips” in *Performance Guide*
- “Index privileges” on page 518
- “Options on the CREATE INDEX statement” on page 261
- “Using an index” on page 260

Related tasks:

- “Dropping an index, index extension, or an index specification” on page 327
- “Renaming an existing table or index” on page 326

Related reference:

- “CREATE INDEX statement” in *SQL Reference, Volume 2*

- “Restrictions on native XML data store” in *XML Guide*

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 *db2advts*.

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 statistics 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. Similarly, large index keys can also 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 64. If you are indexing a typed table, the maximum number of columns is 63. The maximum length of an index

key must not be greater than the index key length limit for the page size. For column stored lengths, see "Byte Counts" in "CREATE TABLE" section in the *SQL Reference*. For the key length limits, see the "Database Manager Limits" table in the *SQL Reference*.

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 might 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 might 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 committed. Such a clean up might 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 environment are built using the same **CREATE INDEX** statement. Data in the indexes is distributed based on the distribution key of the table. When this is done, a B+ tree is created on each database partition in the database partition group. Each B+ tree indexes the part of the table belonging to that database partition. Columns in a unique index defined on a multi-partition database must be a superset of the columns in the distribution key.

Related concepts:

- “User-defined extended index types” on page 265
- “Index privileges” on page 518
- “Using an index” on page 260
- “Options on the CREATE INDEX statement” on page 261
- “Indexes” in *SQL Reference, Volume 1*

Related tasks:

- “Enabling parallelism when creating indexes” on page 10
- “Creating an index” on page 256
- “Dropping an index, index extension, or an index specification” on page 327
- “Renaming an existing table or index” on page 326

Related reference:

- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “SQL and XQuery limits” in *SQL Reference, Volume 1*
- “CREATE INDEX EXTENSION statement” in *SQL Reference, Volume 2*
- “CREATE INDEX statement” in *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:

- “Data access through index scans” in *Performance Guide*
- “Relational index planning tips” in *Performance Guide*
- “Relational index performance tips” in *Performance Guide*
- “Table and index management for MDC tables” in *Performance Guide*
- “Table and index management for standard tables” in *Performance Guide*

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 might 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 might 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:

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 might 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 might 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 an index has the following key values:

```
(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 $1 \leq x \leq m$ 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 SYSTEM(-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 query compilation degree.

In multiple partition databases, unique indexes must be defined as supersets of the distribution key.

Related concepts:

- “Index reorganization” in *Performance Guide*
- “Online index defragmentation” in *Performance Guide*
- “Relational index performance tips” in *Performance Guide*
- “Table and index management for MDC tables” in *Performance Guide*
- “Table and index management for standard tables” in *Performance Guide*

Related tasks:

- “Changing table attributes” on page 298

Related reference:

- “CREATE INDEX statement” in *SQL Reference, Volume 2*
- “dft_degree - Default degree configuration parameter” in *Performance Guide*
- “intra_parallel - Enable intra-partition parallelism configuration parameter” in *Performance Guide*
- “max_querydegree - Maximum query degree of parallelism configuration parameter” in *Performance Guide*

User-defined extended index types

To support user-defined index types, the DB2 database manager 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 might 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:

- “Defining an index extension - example” on page 268

- “Index exploitation” on page 267
- “Index maintenance” on page 266
- “Relational index searching” on page 266

Creating user-defined extended index types

This section discusses the various aspects required when creating your own extended index type.

Index maintenance

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.

You define two of the components that make up the operations of an index through the CREATE INDEX EXTENSION statement.

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:

- “User-defined extended index types” on page 265

Related reference:

- “CREATE INDEX EXTENSION statement” in *SQL Reference, Volume 2*

Relational index searching

Relational 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 relational 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 relational 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 relational 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:

- “User-defined extended index types” on page 265
- “Index exploitation” on page 267
- “Index maintenance” on page 266

Related reference:

- “CREATE INDEX EXTENSION statement” in *SQL Reference, Volume 2*

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 might 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 might 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:

- “Defining an index extension - example” on page 268
- “User-defined extended index types” on page 265
- “Index maintenance” on page 266
- “Relational index searching” on page 266

Related reference:

- “CREATE FUNCTION (External Scalar) statement” in *SQL Reference, Volume 2*

Defining an index extension - example

An example of defining an index extension:

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 data type, ...)
  FROM SOURCE KEY (parm_name data type)
  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 data type, ...)
...
WITH TARGET KEY
  WHEN method_name (parm_name data type, ...)
  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 might

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:

- “User-defined extended index types” on page 265
- “Index exploitation” on page 267
- “Index maintenance” on page 266
- “Relational index searching” on page 266

Related reference:

- “CREATE FUNCTION (External Scalar) statement” in *SQL Reference, Volume 2*
- “CREATE INDEX EXTENSION statement” in *SQL Reference, Volume 2*

Showing related objects

Use the Show Related notebook to show the relationships between tables, indexes, views, aliases, triggers, table spaces, UDFs, and UDTs.

Only first-level dependencies are shown in the table in the Show Related notebook. For example, if you select a table as the target object and the table has a dependency on a view which also has a dependency on another view, only the table dependency is shown. To see the dependency on the view, right-click the view-related object and click **Show Related** in the pop-up menu.

If you want to compare the relationships between several target objects and their related objects, you can open several Show Related notebooks from the Control Center object tree.

There are several reasons to show related objects, including:

- To see the structure of the database.
- To drop or recreate an object, which requires identifying all of the dependency relationships on the object to be dropped or created. Dropping or recreating an object that has one or more dependencies might present problems. For example, if you drop a table with dependent views, the views will be marked inoperative.

To see the SQL query that defines the relationships between objects, click **Show SQL**.

To open the Show Related notebook::

1. From the Control Center, expand the object tree until you find the object you want to work with such as a table, index, view, alias, trigger, tablespace, UDF, or UDT.
2. Right click the object, and click **Show Related** from the pop-up menu. The Show Related window opens.

3. Optional. Repeat the above steps to open another Show Related notebook. You can open multiple Show Related notebooks to show related objects.

To open another view in the Show Related notebook::

From each page of the Show Related notebook, right click the object for which you want to view related objects, and click **Show Related** in the pop-up menu. The Target object changes to the object you just selected. The Show Related notebook page changes to show the objects related to your latest selection. Your previous target object is added to the list.

Note: You can perform other actions in the Show Related notebook by right-clicking the object in the Show Related notebook and clicking an action in the pop-up menu.

Related concepts:

- “Control Center overview” on page 376

Validating related objects

You can customize the SQL statements associated with a selected object. The Related Objects window in the Control Center allows you to identify SQL statements that need to be corrected before the specified table changes are actually applied.

Validation testing checks that all relationships for the selected object are valid, that the necessary user privileges are held, and that data transformations can occur without errors. If invalid statements are found, you can correct the statements.

Prerequisites:

To validate related objects when changing table columns, you must have DBADM authority.

Procedure:

To validate objects:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the Columns page, perform one of the following actions to enable the Related objects button:
 - Rename a column
 - Drop a column
 - Change the data type of a column
 - Change the length, scope, or precision values for a column
 - Change whether a column is nullable
3. Click **Related objects**. The Related Objects window opens.
4. Click **Test All** to test the validity of the SQL statements for all of the objects listed in the **Impacted objects** table. The **Validity** column of the table is updated to indicate if each object is valid or invalid.

- If no objects are invalid, then changes are not needed. Click **Close** to return to the Alter Table notebook.
- If any of the objects is shown to be invalid, select the object in the **Impacted objects** table. The SQL statement for the object will appear in the **SQL statement used for selected object** field.
 - Use the error message box near the bottom of the window to identify the source of any errors for the selected object. Make any needed changes to the associated SQL statement. Click **Apply** to update the statement in the **Impacted objects** table.
 - Click **Test SQL** to check the validity of the corrected SQL statements for the selected object. In the **Validity** column of the **Impacted objects** table, the cell for the selected object is updated to indicate if the object is valid or invalid. If the object is valid, no further changes are needed. If the object is invalid, repeat the previous step to make further changes.

Sometimes an SQL statement cannot be tested individually, as it might rely on other objects to be dropped or created. In this case, click **Test All** to test the validity of the SQL statements for all of the objects listed in the **Impacted objects** table.
- 5. Once any necessary changes have been made and all objects in the **Impacted objects** table are valid, click **Close** to return to the Alter Table notebook.

Related tasks:

- “Showing related objects” on page 270

Estimating space requirements for tables and indexes

Use the Estimate Size window to estimate the amount of storage space required for a new or existing table or index. The estimated storage space is determined from the definition of a particular table and its dependent indexes. Minimum and maximum storage values are estimated for these objects.

Reasons to estimate space requirements for tables and indexes include:

- To create a new table and determine the size of the table space required.
- To create a new table based on the size estimate of an existing table.
- To rearrange a table space. For example, you want to move a table from table space A to table space B, and you want to know how much space the table uses in table space A.
- To identify the amount of space occupied by different objects in the table space because the system is running out of storage space.

Note: For DB2 Enterprise Server Edition databases, size estimates are based on the logical size of the data in the table instead of by database partition.

To open the Estimate Size window, from the Control Center, expand the object tree until you find the **Tables** folder or **Indexes** folder. Click the **Tables** folder or **Indexes** folder. Any existing tables or indexes are displayed in the pane on the right side of the window (contents pane).

- For an existing table or index, right-click the table or index you want in the contents pane, and click **Estimate Size** in the pop-up menu.
- When you are creating a table or index, right-click the table or index folder and click **Create Table** or **Create Index** in the pop-up menu. The Create Table wizard

or Create Index window opens. Complete the information required to create a table or index. Select **Estimate Size** from the Create Table wizard or Create Index window.

Since every row added to the table affects the storage used by the indexes as well as tables, the table and all its related indexes are displayed in the Estimate Size window.

If recent statistics are not available for the table or index, click **Run statistics** before updating the **New total number of rows** and **New average row length** fields and running a size estimate. The calculation of the size estimate can then be based on more accurate information. In the Run Statistics window that opens, either accept the default or select a different value for **New total number of rows**.

Related concepts:

- “Space requirements for temporary tables” in *Administration Guide: Planning*
- “Space requirements for database objects” in *Administration Guide: Planning*
- “Space requirements for indexes” in *Administration Guide: Planning*
- “Space requirements for system catalog tables” in *Administration Guide: Planning*
- “Space requirements for user table data” in *Administration Guide: Planning*

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 279

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 the DB2 database manager is updated by installing a Program Temporary Fix (PTF) or a patch, all the existing DB2 database 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 34

Related tasks:

- “Removing instances” on page 278
- “Updating instance configuration on UNIX” on page 276

Related reference:

- “db2iupdt - Update instances command” in *Command Reference*

Details on changing instances

Before changing an instance, you should list all of the existing instances.

Updating instance configuration on UNIX

Running the **db2iupdt** command updates the specified instance by performing the following:

- Replaces the files in the `sql1ib` 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 `sql1ib` subdirectory under the instance owner’s home directory.

Procedure:

The **db2iupdt** command is found in `/usr/opt/db2_09_01/instance/` directory on AIX. The **db2iupdt** command is found in `/opt/IBM/db2/V9.1/instance/` directory on HP-UX, Solaris, 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 might 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 Server Edition default instance type)
- Database Server with local and remote clients (DB2 Enterprise Server Edition default instance type)
- Client (DB2 client default instance type)

Examples:

- If you installed DB2 Workgroup Server Edition or DB2 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 Server 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 278

Related reference:

- “db2ilist - List instances command” in *Command Reference*
- “db2iupdt - Update instances command” in *Command Reference*

Updating instance configuration on Windows

Running the **db2iupdt** command updates the specified instance by performing the following:

- Replaces the files in the `sql1lib` 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 `sql1lib` subdirectory under the instance owner’s home directory.

Procedure:

The **db2iupdt** command is found in `\sql1lib\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 computer.

- /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 database partitions.
- /u: username,password
Specifies the account name and password for the DB2 service.

Related tasks:

- “Listing instances” on page 41
- “Removing instances” on page 278
- “Updating instance configuration on UNIX” on page 276

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/sql11ib directory (where INSTHOME is the home directory of the instance owner). For example, you might want to save the database manager configuration file, db2system, 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 might be used for other purposes.

The **db2idrop** command removes the instance entry from the list of instances and removes the `sql1ib` 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 `sql1ib` 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 `sql1ib` subdirectory.

Related reference:

- “`db2idrop` - Remove instance command” in *Command Reference*
- “`db2ilist` - List instances command” in *Command Reference*
- “`db2stop` - Stop DB2 command” in *Command Reference*
- “STOP DATABASE MANAGER command” in *Command Reference*
- “TERMINATE command” in *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.

In Version 9.1, the Configuration Advisor is automatically invoked when you create a database. To disable this feature, or to explicitly enable it, use the **db2set** command before creating the database. Examples:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

See Automatic features enabled by default for other DB2 features that are enabled by default.

Prerequisites:

If you plan to change any database partition groups (adding or deleting database partitions, or moving existing database 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.

To access the Configuration Advisor from a client application, call the db2AutoConfig API. To update individual parameters in the database manager configuration or database configuration file from a client application, call the db2CfgSet API.

Related concepts:

- “Benchmark testing” in *Performance Guide*
- “Automatic features enabled by default” in *Administration Guide: Planning*

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*
- “Changing the database configuration across multiple database partitions” on page 281

Related reference:

- “GET DATABASE MANAGER CONFIGURATION command” in *Command Reference*
- “UPDATE DATABASE MANAGER CONFIGURATION command” in *Command Reference*

Changing the database configuration across multiple database partitions

Procedure:

When you have a database that is distributed across more than one database 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 130

Related tasks:

- “Changing node and database configuration files” on page 279

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.

Altering a database partition group

Procedure:

To alter a database partition group using the Control Center:

1. Open the Alter Database Partition Group wizard. To open the Alter Database Partition Group wizard: From the Control Center, expand the object tree until you find the **Database Partition Groups** folder. Click the **Database Partition Groups** folder. Any existing database partition groups are displayed in the contents pane on the right. Right-click the database partition group you want to change and select **Alter** from the pop-up menu. The Alter Database Partition Group wizard opens.
You can also open the Alter Database Partition Group window from the Storage Management view. To open the Storage Management view: From the Control Center window, expand the object tree until you find the database, database partition group, or table space you want to examine in the Storage Management view. Right-click the desired database and select **Manage Storage** from the pop-up menu. The Storage Management view opens.
Note: The first time you launch the Storage Management view from an object you will need to specify your settings in the Storage Management Setup launchpad.
2. Complete each of the applicable wizard pages. The **Finish** push button is enabled when you complete enough information for the wizard to alter the database partition group.

To alter a database partition group using the command line processor: use the **REDISTRIBUTE DATABASE PARTITION GROUP** command.

Once you add or drop database partitions, you must redistribute the current data across the new set of database partitions in the database partition group.

Related concepts:

- “Data redistribution” in *Performance Guide*
- “Management of database server capacity” on page 29

Related tasks:

- “Redistributing data across database partitions” in *Performance Guide*

Related reference:

- “REDISTRIBUTE DATABASE PARTITION GROUP command” in *Command Reference*

Managing database partitions from the Control Center

You can work with database partitions using the Database Partitions view of the Control Center.

Using the Database Partitions view you can restart a database partition, take a database partition out of the rollforward pending state, backup a database partition, restore a database partition, or configure a database partition using the Configuration Advisor.

Authorities:

To work with database partitions you will need authority to attach to an instance. Anyone with SYSADM or DBADM authority can grant you with the authority to access a specific instance.

To configure a database partition or take a database partition out of the rollforward pending state you must have SYSADM, SYSCTRL, or SYSMAINT authority.

Procedure:

To open the Database Partitions view from the Control Center:

1. From the Control Center, expand the object tree until you find the partitioned database for which you want to view the database partitions.
2. Right-click on the partitioned database you want and select Open Database Partitions from menu list.
3. The Database Partitions view opens for the selected partitioned database.

To configure a database partition:

1. Select the database partitions that you want from the Database Partitions view.
2. Select Database Partitions, then Configuration Advisor from the list.
3. The Configuration Advisor opens. Use the Configuration Advisor to specify values for the database configuration parameters.

Related concepts:

- “Adding database partitions in a partitioned database environment” on page 123

- “Database partition and processor environments” in *Administration Guide: Planning*

Related tasks:

- “Adding a database partition server to an instance (Windows)” on page 144
- “Adding a database partition to a running database system” on page 119
- “Changing the database configuration across multiple database partitions” on page 281

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 database partitions or on a single database partition.
- Enable self tuning for a buffer pool, allowing DB2 to adjust the size of the buffer pool in response to your workload.
- 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:

To alter a buffer pool using the Control Center:

1. Open the Alter Buffer Pool window: From the Control Center, expand the object tree until you find the **Buffer Pools** folder. Click on the **Buffer Pools** folder. Any existing buffer pools are displayed in the pane on the right side of the window. Right-click the buffer pool you want and select **Alter** from the pop-up menu. The Alter Buffer Pool window opens.
2. To change the size of a buffer pool, type a new value.
3. Optional: Specify whether to use the default buffer pool size.
4. Optional: Specify whether to alter the buffer pool immediately (this is the default setting), or whether to alter it the next time that the database is restarted.

To alter a buffer pool using the command line:

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; the DB2 database system 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 concepts:

- “Self tuning memory” in *Performance Guide*

Related tasks:

- “Creating a buffer pool” on page 166

Related reference:

- “ALTER BUFFERPOOL statement” in *SQL Reference, Volume 2*

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 want.

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.

Procedure:

To alter a table space using the Control Center:

1. open the Alter Table Space notebook: From the Control Center, expand the object tree until you find the **Table Spaces** folder. Click on the **Table Spaces** folder. Any existing table spaces are displayed in the pane on the right side of the window. Right-click on the table space you want in the contents pane and select **Alter** from the pop-up menu. The Alter Table Space notebook opens.
2. Optional: Change the comment.
3. Optional: Specify the name of the buffer pool in which this table space should reside. The page size of the buffer pool you select must be equal to the page size of the table space.

To alter a table space using the command line, use the **ALTER TABLESPACE** statement.

Related tasks:

- “Adding a container to a DMS table space” on page 285
- “Adding a container to an SMS table space on a database partition” on page 289
- “Dropping a system temporary table space” on page 292

- “Dropping a user table space” on page 291
- “Dropping a user temporary table space” on page 293
- “Modifying containers in a DMS table space” on page 286
- “Renaming a table space” on page 290

Related reference:

- “ALTER TABLESPACE statement” in *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

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 might 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 might 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.

Procedure:

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 Linux and UNIX 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:

- “How containers are added and extended in DMS table spaces” in *Administration Guide: Planning*
- “Table space impact on query optimization” in *Performance Guide*

Related tasks:

- “Adding a container to an SMS table space on a database partition” on page 289

Related reference:

- “ALTER TABLESPACE statement” in *SQL Reference, Volume 2*

Modifying containers in a DMS table space

You can resize the containers in a DMS table space (that is, one created with the MANAGED BY DATABASE clause).

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:

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 might 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 Linux and UNIX 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 might 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 might 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 *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 might be a noticeable degradation in the performance of the database.

The DB2 database manager is set up so that the automatic prefetch size is the default for any table spaces created using Version 8.2 (and later). The DB2 database manager uses the following formula to calculate the prefetch size for the table space:

$$\text{prefetch size} = (\text{number of containers}) \times (\text{number of physical spindles per container}) \times \text{extent size}$$

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. The DB2 database manager 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:

- “Adding a container to a DMS table space” on page 285
- “Altering a table space” on page 284
- “Modifying containers in a DMS table space” on page 286

Adding a container to an SMS table space on a database partition

Restrictions:

You can only add a container to a SMS table space on a database 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 (<database partition_number>)
```

The database partition specified by number, and every partition in the range of database partitions, must exist in the database partition group on which the table space is defined. A database partition_number might 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 database 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 285
- “Modifying containers in a DMS table space” on page 286

Related reference:

- “ALTER TABLESPACE statement” in *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.

To rename a table space using the Control Center:

1. Open the Rename Table Space window: From the Control Center, expand the object tree until you find the **Table Spaces** folder. Click on the **Table Spaces** folder. Any existing table spaces are displayed in the pane on the right side of the window. Right-click on the table space you want and select **Rename** from the pop-up menu. The Rename Table Space window opens.
2. Type a new name for the table space.

To rename a table using the command line, use the **RENAME TABLESPACE statement**.

Related reference:

- “RENAME TABLESPACE statement” in *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 *SQL Reference, Volume 2*

Dropping a user table space

When you drop a user table space, you delete all the data in that table space, free the containers, remove the catalog entries, and cause all objects defined in the table space to be either dropped or marked as invalid.

You can reuse the containers in an empty table space by dropping the table space, but you must COMMIT the DROP TABLESPACE command before attempting to reuse the containers.

You can drop a user table space that contains all of the table data including index and LOB data within that single user table space. You can also drop a user table space that might have tables spanned across several table spaces. That is, you might 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.

Procedure:

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 292
- “Dropping a user temporary table space” on page 293

Related reference:

- “COMMIT statement” in *SQL Reference, Volume 2*
- “DROP statement” in *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 want to add a container to it, and it is an SMS table space, you must first add a new 4 KB page size 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.)

The default table space page size is 4 KB.

Procedure:

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 291
- “Dropping a user temporary table space” on page 293

Related reference:

- “CREATE TABLESPACE statement” in *SQL Reference, Volume 2*
- “DROP statement” in *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 system temporary table space” on page 292
- “Dropping a user table space” on page 291

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*

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.

To drop a database from a client application, call the sqledrpd API. To drop a database at a specified database partition server, call the sqledpan API.

Related reference:

- “DROP DATABASE command” in *Command Reference*
- “GET SNAPSHOT command” in *Command Reference*
- “LIST ACTIVE DATABASES command” in *Command Reference*

Dropping a schema

Before dropping a schema, all objects that were in that schema must be dropped themselves or moved to another schema. The schema name must be in the catalog when attempting the DROP statement; otherwise an error is returned.

Procedure:

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> RESTRICT
```

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, and it must be specified.

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*
- “ADMIN_DROP_SCHEMA procedure – Drop a specific schema and its objects” in *Administrative SQL Routines and Views*

Chapter 6. Altering tables and other related table objects

This section describes tasks that are required for modifying the structure and content of the table and related table objects.

Note that you cannot alter triggers for tables; you must drop any trigger that is no longer appropriate (see “Dropping a trigger” on page 329), and add its replacement (see “Creating triggers” on page 240).

Modifying tables

This section discusses various aspects of modifying tables, including space value compression. It covers how to change table attributes and properties, columns and rows, and keys and constraints.

Space value 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 might 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 table space. For example, the allowable row length is 4005 bytes in a table space 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:

- “Data row compression” on page 188
- “Space compression for tables” on page 187
- “Space value compression for new tables” on page 187

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Copying tables

A basic COPY performs a simple copy of one table to another. This action is a one-time copy only. A new table is defined based on the definition of the selected table, and the contents are copied to the new table. Options that are not copied include:

- Check constraints
- Column default values
- Column comments
- Foreign keys
- Logged and compact option on BLOB columns
- Distinct types

A new table is defined based on the definition of the selected table, and the contents are copied to the new table. You can copy a table into the same database or a different database.

Prerequisites:

To copy a table, you need both:

- One of the following authorities on the source table:
 - SELECT privilege
 - SYSADM or DBADM authority
- One of the following authorities on the target database:
 - CREATETAB and CONNECT privileges, and either:
 - IMPLICIT_SCHEMA authority on the database if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema if the schema name of the table exists
 - SYSADM or DBADM authority

Procedure:

To copy a table using the Control Center:

1. Open the Copy Table window: From the Control Center, expand the object tree until you find the **Tables** folder. Click on the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want to copy and select **Copy** from the pop-up menu. The Copy Table window opens.
2. Specify the name of an existing host or server for the target table and the instance that contains the database that will contain the target table.
3. Specify the database that will contain the target table, the schema for the target table, and a unique name for the target table. If a table with the same name already exists in the schema, the copy will fail.
4. Optional: Select a table space for the target table. Select a REGULAR DMS table space other than the default table space if you want to specify an index table space or long data table space.
5. Optional: Select a table space in which to create any indexes on the target table.
6. Optional: Select a table space in which to store the values of any long columns in the target table.

When you click **OK**, the table that you selected is copied to the target table.

To copy a table using the command line, use the **EXPORT** and **IMPORT** commands.

Related concepts:

- “About databases” in *Administration Guide: Planning*
- “About systems” in *Administration Guide: Planning*
- “Instance creation” on page 34
- “Replicated materialized query tables” in *Administration Guide: Planning*

Related reference:

- “EXPORT command” in *Command Reference*
- “IMPORT Command” in *Command Reference*

Altering a table

Use the Alter Table notebook or the **ALTER TABLE** statement to alter the row format of table data.

Prerequisites:

To alter a table, you must have one of the following authorities or privileges:

- ALTER privilege
- CONTROL privilege
- SYSADM authority
- DBADM authority
- ALTERIN privilege on the table schema

Procedure:

To alter a table using the Control Center:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table that you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
 2. Specify the required information to do the following:
 - Change table properties
 - Add new columns or change existing columns
 - Define new primary keys or change existing primary keys
 - Add new foreign keys or change existing foreign keys
 - Add new check constraints or change existing check constraints
 - Add new partitioning keys or change existing partitioning keys
 - Manage table partitions
- For more information, refer to the online help.

To alter a table using the command line, use the **ALTER TABLE** statement.

To alter a table using a stored procedure, use the **ALTOBJ** procedure.

Related concepts:

- “Using a stored procedure to alter a table” on page 324
- “Using the ALTER TABLE statement to alter columns of a table” on page 300

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “ALTOBJ procedure” in *Administrative SQL Routines and Views*

Changing table attributes

You might 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. For partitioned tables, this lock strategy is applied to both the table lock and the data partition locks for any data partitions accessed. Use of table level locks might improve the performance of queries by limiting the number of locks that need to be acquired and released.

For multidimensional clustering (MDC) tables, using the BLOCKINSERT value for LOCKSIZE causes block-level locking to occur during INSERT operations and

row-level locking during other operations. Block-level locking is useful for transactions that do large insert operations into cells where different transactions make insertions into distinct cells.

For example, after an `ALTER TABLE ... LOCKSIZE BLOCKINSERT` operation, insertions into MDC tables usually cause block locking and not row locking. The only row-level locking that occurs is the next-key locking. This locking is required when the insertion of a record's key into a RID index must wait for a repeatable-read (RR) scan to commit or roll-back before proceeding. This process maintains RR semantics. This option should not be used when there might be multiple transactions inserting data into the same cell concurrently. Exceptions can occur when each transaction has sufficient data to insert per cell and you are not concerned that separate blocks are used for each transaction. In this case, there will be some partially-filled blocks for the cell. This situation causes the cell to be larger than it would otherwise be.

By specifying `APPEND ON`, you can improve the overall performance of the table. Using this option 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:

- “Factors that affect locking” in *Performance Guide*
- “Preventing lock-related performance issues” in *Performance Guide*
- “Lock attributes” in *Performance Guide*
- “Locks and concurrency control” in *Performance Guide*
- “Lock granularity” in *Performance Guide*

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Changing table properties

After you have created a table, you can change the comment, select a lock size, determine the percentage of free space on each page, select data capture for propagation, extend the data capture to include long variable length columns, and indicate whether data is to be appended to the end of the table data.

Prerequisites:

To alter a table, you must have at least one of the following privileges on the table to be altered:

- ALTER privilege
- CONTROL privilege
- SYSADM or DBADM authority
- ALTERIN privilege on the schema of the table

Note: To change the definition of an existing column (in a database that is Version 8.2 or greater), you must have DBADM authority.

Procedure:

To alter table properties using the Control Center:

1. Expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the Table tab:
 - Type a new comment or edit the existing comment.
 - Select a lock size to specify the use of row locks or table locks when accessing the table. Use of the **Lock size** does not prevent normal lock escalation.
Attention: Your new lock size selection is saved in the system but will not be displayed the next time you view this field.
 - Select a value to change the percentage of each page to be left as free space during load or reorganization.
 - Indicate whether extra information regarding changes to this table will be written to the log if this table is replicated.
 - Indicate whether to extend the data capture for propagation function to include long varchar and long vargraphic columns in the log.
Attention: You must first select the **Data capture for propagation** check box to enable the **Include long variable length columns** check box.
 - Indicate whether data is to be appended to the end of the table data.
 - Indicate to the optimizer that the cardinality of your table can vary significantly at run time.
 - Specify how index build logging should be performed:
 - **No change** specifies that no change will be made to the LOG INDEX BUILD table attribute
 - **NULL** specifies that the amount of index build information logged for this table will depend on the value of the LOGINDEXBUILD database configuration parameter.
 - **ON** specifies that enough index build data will be logged for this table to reconstruct indexes during DB2 rollforward or HADR log replay.
 - **OFF** specifies that minimal index build data will be logged.

To alter table properties using the command line, use the **ALTER TABLE** statement.

Related concepts:

- “Primary keys” in *Administration Guide: Planning*

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Altering columns and rows

This section describes how to alter columns and rows on tables and views.

Using the **ALTER TABLE** statement to alter columns of a table

Before creating your database, you need to consider the type and organization of the data that you want to store in it. You need to plan what kind of data your business might need to use and how your business will use that data. However, things change. Despite good planning, there might be new requirements that necessitate changes to the tables in the database. You can use the ALTER TABLE

statement to alter the row format of table data: dropping columns, changing the types of columns, and certain other column attributes.

Restrictions on table access after ALTER TABLE statements containing REORG-recommended operations:

It is important that you plan the implementation of the table alterations well. Perhaps the most important thing to realize when running an ALTER TABLE statement containing a REORG-recommended operation is that once the ALTER TABLE statement has executed, the table will be placed in the Reorg Pending state. This means that the table is inaccessible for almost all operations until you perform a REORG. See the ALTER TABLE statement in the *SQL Reference* for the complete list of ALTER TABLE operations, some of which are also called REORG-recommended operations.

After an ALTER TABLE statement containing REORG-recommended operations, you can execute only the following statements on a table:

```
REORG TABLE
DROP TABLE
ALTER TABLE
RENAME TABLE
TRUNCATE TABLE
```

To allow data recovery in case of a REORG failure, table data might be read using scan-based read only statements, that is, using TABLE SCAN statements. In addition, index-based table access is not allowed. If a table scan-based access is used instead of index-based access, you can also issue a SELECT statement from the table.

The following ALTER TABLE statements require row data validation, and are not allowed following a REORG-recommended ALTER. However, you can execute most of the other ALTER TABLE statements. The ALTER TABLE statements that you *cannot* use are those that require scanning of column data to verify the validity of the alteration operations. Specifically, this means that you cannot execute the following statements on a table:

```
ADD UNIQUE CONSTRAINT
ADD CHECK CONSTRAINT
ADD REFERENTIAL CONSTRAINT
ALTER COLUMN SET NOT NULL
ALTER TABLE ADD REFERENTIAL CONSTRAINT
ALTER TABLE ADD CONSTRAINT
ALTER TABLE ADD UNIQUE CONSTRAINT
```

Examples of ALTER TABLE statements containing REORG-recommended operations:

In addition to placing restrictions on table access after an ALTER TABLE statement containing REORG-recommended operations, the DB2 database manager allows you to specify only three REORG-recommended operations before you perform a classic REORG and before additional REORG-recommended operations will succeed. For this reason, you should code each ALTER TABLE statement containing REORG-recommended operations to change the attributes of as many columns as possible. For example, if you specify the following sequence of ALTER TABLE statements containing only one REORG-recommended operation in each, you will be unable to specify any subsequent ALTER TABLE statements that would require a new row format until you perform a classic REORG:

```
ALTER TABLE foo DROP COLUMN C1
ALTER TABLE foo DROP COLUMN C2
ALTER TABLE foo DROP COLUMN C3
```

You could, however, replace the three ALTER TABLE statements with a single one:

```
ALTER TABLE foo DROP COLUMN C1 DROP COLUMN C2 DROP COLUMN C3
```

Since you can alter only one attribute per column in a single SQL statement—for example, type or nullability—it is possible that changing a column to a new format could require the use of more than one ALTER TABLE statement containing REORG-recommended operations. In such a case, it is important that the order of alterations not allow one alteration to preclude another due to the Reorg Pending state. This means that you should perform operations requiring table data access using the first ALTER TABLE statement containing REORG-recommended operations. For example, if column C1 is an integer and is NULLABLE and you want to change this column to be a NOT NULLABLE BIGINT, the following sequence will fail:

```
ALTER TABLE bar ALTER COLUMN C1 SET DATA TYPE BIGINT
ALTER TABLE bar ALTER COLUMN C1 SET NOT NULL
```

The reason for the failure is that the second ALTER TABLE statement requires a scan of the column C1 to see whether any rows contain the value NULL. Since the table is placed in Reorg Pending state after the first statement, the scan for the second statement cannot be performed.

However, the following sequence will succeed because the first statement does not access the data and does not put the table in Reorg Pending state:

```
ALTER TABLE bar ALTER COLUMN C1 SET NOT NULL
ALTER TABLE bar ALTER COLUMN C1 SET DATA TYPE BIGINT
```

You can perform many operations that alter a table that do not constitute REORG-recommended operations regardless of the number of REORG-recommended operations that you have specified. These include:

```
ADD COLUMN
ALTER COLUMN DEFAULT VALUE
RENAME TABLE
ALTER COLUMN SET DATA TYPE VARCHAR/VARGRAPHIC/CLOB/BLOB/DBCLOB
```

Concurrency during ALTER TABLE execution:

Any ALTER TABLE statement requires exclusive access to a table, as it modifies in-memory structures. For certain statement options in particular, ALTER TYPE and DROP COLUMN—rows in table catalogs will be locked exclusively for UPDATE or DELETE. For this reason, once the ALTER TABLE statement completes, it is important that the unit of work containing the statement be committed or rolled back as soon as possible.

ALTER TABLE authority considerations:

After using an ALTER TABLE statement containing REORG-recommended operations, you must use the classic REORG TABLE statement to make the table accessible again. Having ALTER authority on the table does not necessarily mean that you have the authority to use the REORG TABLE statement; you must have REORG authority.

CASCADE versus RESTRICT semantics when dropping columns:

When dropping a column, DB2 must ensure that any database objects that are dependent on that column—for example, views, triggers, and indexes—are also updated. Two options are available when you specify an ALTER TABLE DROP COLUMN statement: CASCADE and RESTRICT. These options affect how dependent database objects are updated.

CASCADE

CASCADE, the default, automatically handles the dropping of database objects that are dependent on the column being dropped. You should use CASCADE only when you feel confident that you understand the full impact of such an operation. If you do not understand the database object dependencies well, using the CASCADE option might result in performance degradation as a result of implicitly dropping an index. Other side effects could include DML failures on views that were marked inoperative or data integrity issues stemming from inoperative triggers. The following objects are implicitly dropped when you use CASCADE:

- Identity attributes
- SQL routines
- Indexes
- Unique constraints
- Triggers
- Foreign key constraints
- Primary key constraints (this will also cause the implicit deletion of any dependent foreign key constraints)
- Check constraints
- Generated column data
- Views
- Packages

RESTRICT

RESTRICT causes the ALTER TABLE statement to fail if any database object other than a package is found to have a dependency on the column being dropped. Often, it is difficult to generate the complete list of dependent objects for a particular column, but it might be desirable to evaluate each object to decide whether a replacement index should be created following a DROP COLUMN operation. In cases such as this, you might want to specify the RESTRICT option and remove or plan for the re-creation of each affected object. The object type and name first detected with a column-level dependency are returned as part of the error message if the ALTER TABLE statement fails.

Related concepts:

- “Using a stored procedure to alter a table” on page 324
- “Table reorganization” in *Performance Guide*

Related tasks:

- “Altering a table” on page 297

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “DROP statement” in *SQL Reference, Volume 2*
- “REORG INDEXES/TABLE command” in *Command Reference*

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 305

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Changing columns (properties)

From the Alter Table notebook, you can use the Change Column window to change the properties for new or existing columns in a table. This window is accessed from the Columns page of the Alter Table notebook. You can also change the column definitions by directly editing the column information on the Columns page of the Alter Table notebook.

If you are altering a pre-version 8.2 database, you can use the Column Properties window to change the comment for existing columns in a table or change the length of an existing VARCHAR column. You can also change the formula that DB2 uses to determine values for a generated column.

Prerequisites:

To alter a table, you must have at least one of the following privileges on the table to be altered:

- ALTER privilege
- CONTROL privilege
- SYSADM or DBADM authority
- ALTERIN privilege on the schema of the table

To change the definition of a existing column, to edit and test SQL when changing table columns, or to validate related objects when changing table columns, you must have DBADM authority.

Procedure:

To change column properties using the Control Center:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want to change and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the Columns page, select a column and click **Change**. The Change Columns or Change Properties window opens.
3. Make the necessary changes. For more information, refer to the online help for this window.

To change column properties using the command line, use the ALTER TABLE statement. For example:

```
ALTER TABLE EMPLOYEE  
  ALTER COLUMN WORKDEPT  
  SET DEFAULT '123'
```

Related concepts:

- “Using the ALTER TABLE statement to alter columns of a table” on page 300

Related tasks:

- “Adding columns to an existing table” on page 304
- “Defining a generated column on a new table” on page 219
- “Defining a generated column on an existing table” on page 321

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Modifying a column definition

You can use the ALTER TABLE statement to:

- Drop a column, using the new DROP COLUMN clause
- Change a column type, using the ALTER COLUMN SET DATA TYPE clause
- Change the nullability attribute of a column, using the SET NOT NULL or the DROP NOT NULL clause

For example, you can increase the length of an existing VARCHAR or VARCHARIC column. The number of characters might increase up to a value dependent on the page size used.

You can also 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.

When changing these table attributes using SQL, it is no longer necessary to drop the table and then recreate it, a time consuming process that can be complex when object dependencies exist.

Procedure:

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 308

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Removing rows from a table or view

You can change the contents of a table or view by deleting rows. Deleting a row from a view deletes the rows from the table on which the view is based. The DELETE statement is used to:

- Delete one or more rows that have been optionally determined by a search condition. This is known as a *searched DELETE*.
- Delete exactly one row that has been determined by the current position of a cursor. This is known as a *positioned DELETE*.

The DELETE statement can be embedded in an application program or issued as a dynamic SQL statement.

Procedure:

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:

- “Factors that affect locking” in *Performance Guide*
- “Preventing lock-related performance issues” in *Performance Guide*
- “Locks and concurrency control” in *Performance Guide*
- “Lock granularity” in *Performance Guide*

Related reference:

- “DELETE statement” in *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 219
- “Defining an identity column on a new table” on page 220

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 occurring, 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 + 1>
```

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:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “MAX aggregate function” in *SQL Reference, Volume 1*
- “LOAD command” in *Command Reference*

Altering keys and constraints

This section describes how to alter table keys and constraints. You can only alter constraints by dropping them and then adding new ones to take their place.

Adding primary keys

Procedure:

To add primary keys using the Control Center:

1. Expand the object tree until you see the **Tables** folder.
2. Right-click the table you want to modify, and select **Alter** from the pop-up menu.
3. On the **Keys** page, select one or more columns as primary keys.
4. Optional: Enter the constraint name of the primary key.

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 310

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*

Changing primary keys

You can change the primary key for the table that you are creating. A primary key is a column or set of columns that can be used to identify or access a particular row or rows. A primary key is a unique key, that is, it is a key that is constrained so that no two of its values are equal. A table cannot have more than one primary key, and the columns of a primary key cannot contain null values.

Prerequisites:

To alter a table with a primary key, you must have at least one of the following privileges on the table to be altered:

- ALTER privilege
- CONTROL privilege
- SYSADM or DBADM authority
- ALTERIN privilege on the schema of the table

Procedure:

To change primary keys using the Control Center:

1. Open Change Primary Key window: From the Control Center, expand the object tree until you find the **Tables** folder. Right-click the **Tables** folder and select **Create** from the pop-up menu. The Create Table wizard opens. On the Keys page, select a primary key in the table and click **Change**. The Change Primary Key window opens.
2. Select the column or columns that you want to define as primary key columns. You can define up to 16 columns to be primary key columns.
3. Optional: Type the constraint name of the primary key.

To change primary keys using the command line, use the **ALTER TABLE** statement.

Related tasks:

- “Adding primary keys” on page 309
- “Dropping primary keys” on page 316

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Adding foreign keys

When a foreign key is added to a table, packages and cached dynamic SQL containing the following statements might be marked as invalid:

- Statements that insert or update the table containing the foreign key
- Statements that update or delete the parent table.

Procedure:

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 **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.
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 the foreign key.

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 366

Related tasks:

- “Adding primary keys” on page 309

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Changing foreign keys

You can change foreign keys for your table or nickname. A foreign key is a column or set of columns in a table or nickname whose values are required to match at least one primary key value of a row of its parent table or nickname. A referential constraint is the rule that the values of the foreign key are valid only if either:

- They appear as values of a parent key (primary key).
- Some component of the foreign key is null.

Prerequisites:

To alter a table with a foreign key, you must have at least one of the following privileges on the table to be altered:

- ALTER privilege
- CONTROL privilege
- SYSADM or DBADM authority
- ALTERIN privilege on the schema of the table

Procedure:

To change foreign keys using the Control Center:

1. Open the Alter Table notebook if you are adding a unique key to a table: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want in the contents pane and select **Alter** from the pop-up menu. The Alter Table notebook opens.
If you are altering a foreign key on a nickname, open the Alter Nickname notebook.
2. On the Keys page, select a foreign key and click **Change**. The Change Foreign Key window opens.
3. Optional: Select a different parent table or nickname.
4. Specify the schema and name of the new parent table or nickname.
5. Optional: Select a new foreign key.
6. Optional: Change the action specified for "on delete" and "on update".
7. Optional: Change the name of the constraint

To change foreign keys using the command line, use the **ALTER TABLE** statement.

Related concepts:

- "Foreign keys in a referential constraint" on page 226

Related tasks:

- "Adding foreign keys" on page 310
- "Dropping foreign keys" on page 316

Adding unique keys

Use the Add Unique Key window to define a unique key for your table or nickname. You can define more than one unique key for the same table or nickname. For tables, these windows are accessed from the Keys page of the Alter Table notebook. For nicknames, these windows are accessed from the Keys page of Alter Nickname notebook.

Procedure:

To add unique keys using the Control Center:

1. Open the Add Unique Key window: From the Control Center, expand the object tree until you find the Tables folder. Click the Tables folder. Any existing tables are displayed in the pane on the right side of the window (the contents pane). Right-click the table you want in the contents pane and select **Alter** from the pop-up menu. The Alter Table notebook opens. If you are adding a unique key to a nickname, open the Alter Nickname notebook. On the Keys page, click **Add**. The Add Unique Key window opens.
2. Select the column or columns that you want to define or change as unique key columns.
3. Optional: Type the constraint name of the unique key.

To add unique keys using the command line, use the **ALTER TABLE** statement.

Related tasks:

- "Changing unique keys" on page 313

Changing unique keys

You can change a unique key for the table or nickname that you are altering. You can define more than one unique key for the same table or nickname.

Procedure:

To change unique keys using the Control Center:

1. Open the Alter Table notebook if you are adding a unique key to a table: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want in the contents pane and select **Alter** from the pop-up menu. The Alter Table notebook opens.
If you are adding a unique key to a nickname, open the Alter Nickname notebook.
2. On the Keys page, select a unique key from the table and click **Change**. The Change Unique Key window opens.
3. Select the column or columns that you want to define as unique key columns.
4. Optional: Type the constraint name of the unique key.

To change unique keys using the command line, use the **ALTER TABLE** statement.

Related tasks:

- “Adding unique keys” on page 312

Adding unique constraints

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.

Procedure:

To add unique constraints using the Control Center:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the Check Constraints page, click **Add**.
3. On the Add Check Constraint window, complete the necessary information.

To define dimensions using the command line, use the **ADD CONSTRAINT** option of the **ALTER TABLE** statement. For example, 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 a table” on page 223

- “Dropping a unique constraint” on page 315

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Adding a table check constraint

Check constraints can be added to an existing table with the ALTER TABLE statement. The constraint name cannot be the same as any other constraint specified within an ALTER TABLE statement, and must be unique within the table (this includes the names of any referential integrity constraints that are defined). Existing data is checked against the new condition before the statement succeeds.

When a table check constraint is added, packages and cached dynamic SQL that insert or update the table might be marked as invalid.

Procedure:

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 **Constraints** page, click **Add**.
4. On the **Add Check Constraint** window, complete the information.

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 366

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*

Changing check constraints

You can change check constraints to the table that you are creating. A check constraint sets restrictions on data added to the table. Check constraints are enforced whenever rows in the table are inserted or updated.

Prerequisites:

To change check constraints, you must have at least one of the following privileges on the table to be altered:

- ALTER privilege
- CONTROL privilege

- SYSADM or DBADM authority
- ALTERIN privilege on the schema of the table

Note: To change the definition of a existing column (in a database that is Version 8.2 or greater), you must have DBADM authority.

Procedure:

To change check constraints using the Control Center:

1. Open the Change Check Constraint window: From the Control Center, expand the object tree until you find the Tables folder. Right-click the Tables folder and select Create from the pop-up menu. The Create Table wizard opens. On the Constraints page, select a constraint in the table and click Change. The Change Check Constraint window opens.
2. Specify the check condition for the constraint that you are changing.
3. Optional: Type a name for the check constraint.
4. Optional: Type a comment to document the check constraint.

To change check constraints using the command line, use the **ALTER TABLE** statement.

Related tasks:

- “Adding check constraints” on page 229

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Dropping a unique constraint

You can explicitly drop a unique constraint using the ALTER TABLE statement. The name of all unique constraints on a table can be found in the SYSCAT.INDEXES system catalog view.

Dropping this unique constraint invalidates any packages or cached dynamic SQL that used the constraint.

Procedure:

To drop a unique constraint using the Control Center:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the Check Constraints page, select the unique constraints that you want to drop, and select **Drop**.

To drop a unique constraint using the command line, use the **ALTER TABLE** statement. The following SQL statement drops the unique constraint NEWID from the EMPLOYEE table:

```
ALTER TABLE EMPLOYEE
DROP UNIQUE NEWID
```

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Dropping primary keys

Procedure:

To drop 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 **Keys** page, select the primary keys to drop.

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 might 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 366

Related tasks:

- “Dropping foreign keys” on page 316

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Dropping foreign keys

Procedure:

To drop 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 **Keys** page, click **Add**.
4. Select the foreign keys at right to drop.

To drop foreign keys 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 366

Related tasks:

- “Dropping primary keys” on page 316

Related reference:

- “ALTER TABLE statement” in *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 **Constraints** page, select the check constraint to drop, click **Remove**.

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 366


Related tasks:

- “Adding a table check constraint” on page 314

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*

Changing distribution keys

 You can only change a distribution key on tables in a single database partition. First drop the existing distribution key, and then create another.

Procedure:

To change distribution keys using the Control Center:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the Keys page, select a distribution key in the table and click **Change**. The Change Distribution Key window opens.
3. Select the columns that you want to add as distribution key columns and move them to the **Selected columns** box.

To change distribution keys using the command line, use the DROP DISTRIBUTION option of the **ALTER TABLE** statement. For example, the following SQL statement drops the distribution key MIX_INT from the MIXREC table:

```
ALTER TABLE MIXREC
  DROP DISTRIBUTION
```

You cannot change the distribution key of a table spanning multiple database partitions. If you try to drop it, an error is returned.

To change the distribution key of multiple database partitions, either:

- Export all of the data to a single database partition and then follow the above instructions.
- Export all of the data, drop the table, recreate the table redefining the distribution 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 distribution key before implementing the design of large databases.

Related concepts:

- “Distribution keys” in *Administration Guide: Planning*

Related reference:

- “ALTER TABLE statement” in *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 *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 320

Related reference:

- “ALTER SEQUENCE statement” in *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 319

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*

Dropping or removing columns

Prerequisites:

Procedure:

To drop or remove columns using the Control Center:

1. Open the Alter Table notebook: From the Control Center, expand the object tree until you find the **Tables** folder. Click the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Alter** from the pop-up menu. The Alter Table notebook opens.
2. On the columns page, select the columns that you want to drop and click **Remove**. If you change your mind before clicking **OK**, you can click **Undo remove**.

To define dimensions using the command line, use the ADD CONSTRAINT option of the ALTER TABLE statement. For example, 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:

- “Adding columns to an existing table” on page 304

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 might 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 VARCHARIC, and user-defined types defined using the same excluded data types.

Generated columns cannot be used in constraints, 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 might 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 set integrity pending state.
`SET INTEGRITY FOR t1 OFF CASCADE DEFERRED`
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 > c2 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 set integrity 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 might 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 set integrity 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 set integrity 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 GENERATED
```

- c. Turn the not logged initially off again by committing the transaction.

```
COMMIT
```

The values for generated columns can also simply be checked by applying the expression as if it is an equality check constraint:

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

If values have been placed in a generated column using LOAD for example, and you know that the values match the generated expression, then the table can be taken out of the set integrity 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 219

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “COMMIT statement” in *SQL Reference, Volume 2*
- “LOCK TABLE statement” in *SQL Reference, Volume 2*
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*
- “UPDATE statement” in *SQL Reference, Volume 2*
- “db2look - DB2 statistics and DDL extraction tool command” in *Command Reference*
- “Restrictions on native XML data store” in *XML Guide*

Declaring a table volatile

A *volatile* table is defined as a table whose contents can vary from empty to very large at run time. The volatility or extreme changeability of this type of table makes reliance on the statistics collected by RUNSTATS inaccurate. Statistics are gathered at, and only reflect, a point in time. To generate an access plan that uses a volatile table can result in an incorrect or poorly performing plan. For example, if statistics are gathered when the volatile table is empty, the optimizer tends to favor accessing the volatile table using a table scan rather than an index scan.

To prevent this, you should consider declaring the table as volatile using the ALTER TABLE statement. By declaring the table volatile, the optimizer will consider using index scan rather than table scan. The access plans that use declared volatile tables will not depend on the existing statistics for that table.

Procedure:

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 *SQL Reference, Volume 2*

Using a stored procedure to alter a table

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 might 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. The DB2 database manager 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 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 might 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 carry out 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 concepts:

- “Using the ALTER TABLE statement to alter columns of a table” on page 300

Related tasks:

- “Altering a table” on page 297

Related reference:

- “Supported functions and administrative SQL routines and views” in *SQL Reference, Volume 1*
- “ALTOBJ procedure” in *Administrative SQL Routines and Views*

Modifying indexes

This section describes how to modify indexes.

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 or

XQuery 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 *SQL Reference, Volume 2*

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 might 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 and XQuery 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 366

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “DROP statement” in *SQL Reference, Volume 2*

Modifying triggers

This section describes how to modify triggers.

Updating view contents using triggers

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, the view to update data in the underlying tables cannot be used 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 triggers” on page 240

Related reference:

- “CREATE TRIGGER statement” in *SQL Reference, Volume 2*

Dropping a trigger

A trigger object can be dropped using the DROP statement, but this procedure will cause dependent packages to be marked invalid, as follows:

- If an update trigger without an explicit column list is dropped, then packages with an update usage on the target table are invalidated.

- If an update trigger with a column list is dropped, then packages with update usage on the target table are only invalidated if the package also had an update usage on at least one column in the column-name list of the CREATE TRIGGER statement.
- If an insert trigger is dropped, packages that have an insert usage on the target table are invalidated.
- If a delete trigger is dropped, packages that have a delete usage on the target table are invalidated.

A package remains invalid until the application program is explicitly bound or rebound, or it is run and the database manager automatically rebinds it.

Related tasks:

- “Creating triggers” on page 240

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*

Modifying aliases and views

This section describes how to modify aliases and views.

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 366

Related tasks:

- “Creating triggers” on page 240
- “Creating a view” on page 251
- “Recovering inoperative views” on page 331

Related reference:

- “ALTER VIEW statement” in *SQL Reference, Volume 2*
- “DROP statement” in *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 330

Related reference:

- “CREATE VIEW statement” in *SQL Reference, Volume 2*
- “DROP statement” in *SQL Reference, Volume 2*
- “GRANT (Table, View, or Nickname Privileges) statement” in *SQL Reference, Volume 2*
- “SYSCAT.VIEWS catalog view” in *SQL Reference, Volume 1*

Dropping aliases

When you drop an alias, its description is deleted from the catalog, any packages and cached dynamic queries that reference the alias are invalidated, and all views and triggers dependent on the alias are marked inoperative.

Prerequisites:

To drop an alias, you must be defined to DB2 as the creator of the alias, or you must have one of the following authorizations:

- SYSADM authority
- DBADM authority on the database in which the alias is stored
- The DROPIN privilege on the alias’s schema

Procedure:

To drop an alias using the Control Center:

1. Expand the object tree until you find the **Alias** folder below the database that contains the alias that you want to drop. Click on the **Alias** folder. Any existing aliases are displayed in the pane on the right side of the window. Right-click the alias that you want to drop and select **Drop** from the pop-up menu. The Confirmation window opens.
2. Confirm the drop request.

To drop aliases using the command line, use the **DROP** statement.

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*

Modifying UDFs and UDTs

This section describes how to modify user-defined functions and user-defined types.

Altering a user-defined structured type

Procedure:

After creating a structured type, you might 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:

- “Structured type hierarchies” in *SQL Guide*
- “User-defined structured types” in *SQL Guide*

Related tasks:

- “Creating structured types” in *SQL Guide*

Related reference:

- “ALTER TYPE (Structured) statement” in *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 *SQL Reference, Volume 2*
- “BIND command” in *Command Reference*
- “PRECOMPILE command” in *Command Reference*
- “REBIND command” in *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 types (UDTs)” on page 246

Related tasks:

- “Creating a type mapping in a federated system” on page 248
- “Creating a user-defined distinct type” on page 247

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*

Modifying materialized query tables

This section describes how to modify materialized query tables.

Altering materialized query table properties

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 set integrity 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.

Procedure:

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 201
- “Dropping a materialized query or staging table” on page 365
- “Refreshing the data in a materialized query table” on page 336

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *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:

- “Altering materialized query table properties” on page 335
- “Creating a materialized query table” on page 201

Related reference:

- “REFRESH TABLE statement” in *SQL Reference, Volume 2*

Modifying partitioned tables

This section describes how to modify partitioned tables.

Altering partitioned tables

All relevant clauses of the ALTER TABLE statement are supported for a partitioned table. In addition, the ALTER TABLE statement allows you to ADD new data partitions, roll-in (ATTACH) new data partitions, and roll-out (DETACH) existing data partitions.

Prerequisites:

To alter a partitioned table to detach a data partition the user must have the following authorities or privileges:

- The user performing the DETACH operation must have the authority needed to ALTER, to SELECT from, and to DELETE from the source table.
- The user must also have the authority needed to create the target table. Therefore, to alter a table to detach a data partition, the privilege held by the authorization ID of the statement must include at least one of the following authorities or privileges on the target table:
 - SYSADM or DBADM authority
 - CREATETAB authority on the database and USE privilege on the table spaces used by the table as well as one of:
 - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema.

To alter a partitioned table to attach a data partition, the privileges held by the authorization ID of the statement must include at least one of the following authorities or privileges on the source table:

- SELECT privilege on the source table and DROPIN privilege on the schema of the source table
- CONTROL privilege on the source table
- SYSADM or DBADM authority.

To alter a partitioned table to add a data partition, the privileges held by the authorization ID of the statement must have privileges to use the table space where the new partition is added, and include at least one of the following authorities or privileges on the source table:

- ALTER privilege
- CONTROL privilege
- SYSADM
- DBADM
- ALTERIN privilege on the table schema

Usage guidelines:

- Each ALTER TABLE statement issued with the PARTITION clause must be in a separate SQL statement.
- No other ALTER operations are permitted in an SQL statement containing an ALTER TABLE...PARTITION operation. For example, you cannot attach a data partition and add a column to the table in a single SQL statement.
- Multiple ALTER statements can be executed, followed by a single SET INTEGRITY statement.

Procedure:

You can alter a table from the DB2 Control Center or from the DB2 command line processor (CLP).

To use the DB2 Control Center to alter a partitioned table :

1. Expand the Table folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click the table that you want to alter and select Open Data Partitions from the list of actions.
3. In the Open Data Partitions window select the button associated with your task. If you are adding, the Add Data Partition window opens. If you are attaching, the Attach Data Partition window opens. If you are detaching the Detach Data Partition window opens.
4. Specify the required fields.

To use the DB2 command line to alter a partitioned table, issue the **ALTER TABLE** statement.

Related concepts:

- “Attributes of detached data partitions” on page 354
- “Data partitions” in *Administration Guide: Planning*

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering a table” on page 297
- “Dropping a data partition” on page 358

- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “Command Line Processor (CLP) samples” in *Samples Topics*

Guidelines and restrictions on altering partitioned tables with attached or detached data partitions

The following section identifies the most common alter table actions and special considerations in the presence of attached and detached data partitions.

Adding or altering a check or foreign key constraint:

Adding a check on a foreign key constraint is supported with attached and detached data partitions.

Adding a column:

When adding a column to a table with attached data partitions, the column is also added to the attached data partitions. When adding a column to a table with detached data partitions, the column is not added to the detached data partitions, because the detached data partitions are no longer physically associated to the table.

Altering a column:

When altering a column in a table with attached data partitions, the column will also be altered on the attached data partitions. When altering a column in a table with detached data partitions, the column is not altered on the detached data partitions, because the detached data partitions are no longer physically associated to the table.

Adding a generated column:

When adding a generated column to a partitioned table with attached or detached data partitions, it must respect the rules for adding any other types of columns.

Adding or modifying an index:

When creating, recreating or reorganizing an index on a table with attached data partitions, the index does not include the data in the attached data partitions because the SET INTEGRITY statement maintains all indexes for all attached data partitions. When creating, recreating or reorganizing an index on a table with detached data partitions, the index does not include the data in the detached data partitions, unless the detached data partition has a detached dependents or staging tables that need to be incrementally refreshed with respect to the data partition. In this case, the index includes the data for this detached data partition.

WITH EMPTY TABLE:

You cannot empty a table with attached data partitions.

ADD MATERIALIZED QUERY AS:

Altering a table with attached data partitions to an MQT is not allowed.

Altering additional table attributes that are stored in a data partition:

The following table attributes are also stored in a data partition. Changes to these attributes are reflected on the attached data partitions, but not on the detached data partitions.

- DATA CAPTURE
- VALUE COMPRESSION
- APPEND
- COMPACT/LOGGED FOR LOB COLUMNS
- ACTIVATE NOT LOGGED INITIALLY (WITH EMPTY TABLE)

Related concepts:

- “Attributes of detached data partitions” on page 354
- “Understanding clustering index behavior on partitioned tables” in *Performance Guide*
- “Data partitions” in *Administration Guide: Planning*
- “Understanding index behavior on partitioned tables” in *Performance Guide*
- “Large object behavior in partitioned tables” in *SQL Reference, Volume 1*
- “Partitioned materialized query table behavior” on page 206
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Altering a table” on page 297
- “Dropping a data partition” on page 358
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*

Rotating data in a partitioned table

Rotating data in a DB2 Database for Linux, UNIX, and Windows refers to a method of reusing space in a data partition by removing obsolete data from the table then adding new data. Table partitioning functionality allows you to detach the data partition with the obsolete data then attach a new data partition with the latest data.

Prerequisites:

To detach a data partition from a partitioned table the user must have the following authorities or privileges:

- The user performing the DETACH operation must have the authority needed to ALTER, to SELECT from, and to DELETE from the source table.
- The user must also have the authority needed to CREATE the target table. Therefore, to alter a table to detach a data partition, the privilege held by the authorization ID of the statement must include at least one of the following authorities or privileges on the target table:
 - SYSADM or DBADM authority
 - CREATETAB authority on the database and USE privilege on the table spaces used by the table as well as one of:
 - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema.

To alter a table to attach a data partition, the user must have the following authorities or privileges:

- The user performing the attach must have the authority needed to ALTER and to INSERT into the target table
- The user must also be able to SELECT from and to DROP the source table. Therefore, to alter a table to attach a data partition, the privileges held by the authorization ID of the statement must include at least one of the following on the source table:
 - SELECT privilege on the source table and DROPIN privilege on the schema of the source table
 - CONTROL privilege on the source table
 - SYSADM or DBADM authority.

Procedure:

You can rotate data in a partitioned table from the DB2 Control Center or from the DB2 command line processor (CLP).

To use the DB2 Control Center to rotate data in a partitioned table:

1. Expand the Table folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click the table that you want to alter, and select Open Data Partitions from the list of actions.
3. In the Open Data Partitions window, click the button associated with your task. If you are attaching, the Attach Data Partition window opens. If you are detaching, the Detach Data Partition window opens.
4. Specify the required fields.

To use the DB2 command line to rotate data in a partitioned table, issue the **ALTER TABLE** statement.

Example:

This example demonstrates how to update the stock table by removing the data from December 2001 and replacing it with the latest data from December 2003.

1. Remove the old data from table *stock*.

```
ALTER TABLE stock DETACH PARTITION  
dec01 INTO newtable;
```

2. Load the new data. Using LOAD with the REPLACE option overwrites existing data.

```
LOAD FROM data_file OF DEL REPLACE INTO newtable
```

Note:If there are detached dependents, then you must run the SET INTEGRITY statement on the detached dependents before you can load the detached table.

3. If desired, perform data cleansing. Data cleansing activities include:

- Filling in missing values
- Deleting inconsistent and incomplete data
- Removing redundant data arriving from multiple sources
- Transforming data
 - Normalization (Data from different sources that represents the same value in different ways must be reconciled as part of rolling the data into the warehouse.)
 - Aggregation (Raw data that is too detailed to store in the warehouse must be pre-aggregated during roll-in.)

4. Attach the new data as a new range.

```
ALTER TABLE stock ATTACH PARTITION dec03  
STARTING '12/01/2003' ENDING '12/31/2003'  
FROM newtable;
```

Attaching a data partition drains queries and invalidates packages.

5. Use the SET INTEGRITY statement to update indexes and other dependent objects. Read and write access is permitted during the execution of the SET INTEGRITY statement.

```
SET INTEGRITY FOR stock ALLOW WRITE ACCESS  
IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

Related concepts:

- “Attributes of detached data partitions” on page 354
- “Data partitions” in *Administration Guide: Planning*
- “Partitioned materialized query table behavior” on page 206
- “Optimization strategies for partitioned tables” in *Performance Guide*
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Altering a table” on page 297
- “Dropping a data partition” on page 358
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “ALTER TABLE statement” in *SQL Reference, Volume 2*

- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*
- “Command Line Processor (CLP) samples” in *Samples Topics*

Examples of rolling in and rolling out partitioned table data

The following examples address a common administration operation in data warehouses where new data is rolled in at the start of each month and old data is potentially rolled out based on a particular date. Example 1 covers the DETACH operation (roll-out) by removing obsolete data from the table. Variations include:

1. deleting the data
2. moving the data to another table

Examples 2 and 3 cover both an ADD operation and an ATTACH operation (roll-in) by loading new data into the table. Variations include:

1. transforming the data, loading it into the non-partitioned table, then attaching the data partition (traditional extract, transform and load (ETL))
2. loading the data into the non-partitioned table, transforming the data, then attaching the data partition

Example 1: Using partitioned tables, the roll-out operation is simply a DETACH operation on the appropriate data partition:

```
ALTER TABLE stock DETACH PART dec01 INTO stock_drop;
DROP TABLE stock_drop;
```

To accelerate the DETACH operation, index cleanup on the source table is done automatically through a background asynchronous index cleanup process. If there are no detached dependents defined on the source table, there is no need to issue the SET INTEGRITY statement to complete the DETACH operation.

Instead of dropping the table as described in the previous example, it is also possible to attach the table to another table, or truncate it and use it as a table to load new data into before reattaching it. You can perform these operations immediately, even before the asynchronous index cleanup has completed, except where the stock table has detached dependents.

To detect that a detached table is not yet accessible, query the SYSCAT.TABDETACHEDDEP catalog view. If any inaccessible detached tables are detected, run the SET INTEGRITY statement with the IMMEDIATE CHECKED option on all the detached dependents to transition the detached table to a regular accessible table. If you try to access a detached table before all its detached dependents are maintained, error code SQL20285N is returned.

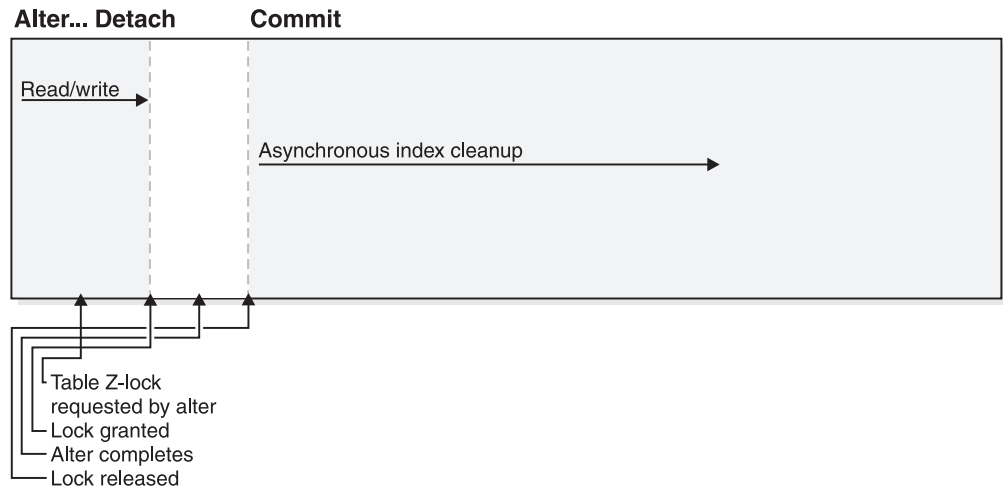


Figure 3. This figure demonstrates the stages of data availability during a DETACH operation. Asynchronous index cleanup commences immediately after the DETACH operation is committed if there are no detached dependents. Otherwise, asynchronous index cleanup commences after the maintenance of the detached dependents is committed.

Rolling in data:

The following example illustrates the steps to load data into a non-partitioned table and then add that data partition to the rest of the table.

Example 2: Create a new, empty range:

```
ALTER TABLE stock ADD PARTITION dec03
STARTING FROM '12/01/2003' ENDING AT '12/31/2003';
```

This ALTER TABLE ...ADD operation drains queries running against the stock table and invalidate packages. That is, existing queries complete normally before the ADD operation continues. Once the ADD operation is issued, any new queries accessing the stock table block on a lock.

Load data into the table:

```
LOAD FROM data_file OF DEL INSERT
INTO stock ALLOW READ ACCESS;
```

Use the SET INTEGRITY statement to validate constraints and refresh dependent materialized query tables (MQTs):

```
SET INTEGRITY FOR stock ALLOW READ
ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
COMMIT WORK;
```

Tip:One advantage for using ALTER TABLE ...ADD PARTITION followed by a LOAD operation versus a LOAD operation followed by ALTER TABLE ...ATTACH is if the table has no constraints or MQTs defined, the SET INTEGRITY statement is not required to make the new data available. There are disadvantages to adding a new data partition and loading data directly into the table. The principal disadvantage of using the ALTER TABLE ...ADD PARTITION statement is that it prevents updates to the table both during the Load operation itself, and during any subsequent SET INTEGRITY statement. While both the ALTER TABLE ...ADD PARTITION statement and the ALTER TABLE ...ATTACH PARTITION statement cause package invalidation, the LOAD command followed by the ALTER ...ATTACH PARTITION operation yields better data availability. However, the

ALTER TABLE ...ADD PARTITION statement followed by the IMPORT command or a regular INSERT statement makes good sense for situations in which the data is not rolled-in in large blocks, but instead trickles in. Adding a data partition also makes sense in cases where the data being rolled in does not match the data partition boundaries.

Rolling in data into a new table:

Example 3: In this example, ATTACH is used to facilitate loading a new range of data into an existing partitioned table. Typically, data is loaded into a new, empty table to perform any necessary cleaning and checking of the data without impacting the target table. After the data is prepared, the newly loaded data partition is attached.

```
CREATE TABLE dec03(. . . . .);
LOAD FROM data_file OF DEL REPLACE INTO dec03;
```

Before rolling in your table data, data cleansing might be required before the data is attached. Data cleansing activities include:

- Filling in missing values
- Deleting inconsistent and incomplete data
- Removing redundant data arriving from multiple sources
- Transforming data
 - Normalization (Data from different sources that represents the same values in different ways must be reconciled as part of rolling the data into the warehouse.)
 - Aggregation (Raw data that is too detailed to store in the warehouse, must be preaggregated during roll-in.)

Next, roll in the data:

```
ALTER TABLE stock ATTACH PARTITION dec03
STARTING FROM '12/01/2003' ENDING AT '12/31/2003'
FROM dec03;
```

During an ATTACH operation, one or both of the STARTING and ENDING clauses must be supplied and the lower bound (STARTING) must be less than or equal to the upper bound (ENDING). In addition, the newly attached data partition must not overlap with an existing data partition range in the target table. If the highest range has been defined as MAXVALUE, then any attempt to attach a new high range fails because it overlaps the existing high range. This restriction also applies to MINVALUE. You cannot add or attach a new data partition in the middle unless it falls in an existing gap in the ranges. Boundaries not specified by the user are determined when the table is created.

The ALTER TABLE ...ATTACH operation drains all queries and invalidate packages dependent on the stock table. That is, existing queries complete normally before the ATTACH operation continues. Once the ATTACH operation is issued, any new queries accessing the stock table block on a lock. The stock table is z-locked (completely inaccessible) during this transition. The data in the attached data partition is not yet visible, because it has not yet been validated by the SET INTEGRITY statement. **Tip:** Issue a COMMIT WORK statement immediately after the ATTACH operation to make the table available for use.

```
COMMIT WORK;
```

The SET INTEGRITY statement is necessary to verify that the newly attached data is in range. It also does any necessary maintenance of indexes and other dependent

objects such as MQTs. Until the SET INTEGRITY statement is committed, the new data is not visible. The existing data in the stock table is fully accessible for both reading and writing if online SET INTEGRITY is used. The default while SET INTEGRITY is running is ALLOW NO ACCESS mode.

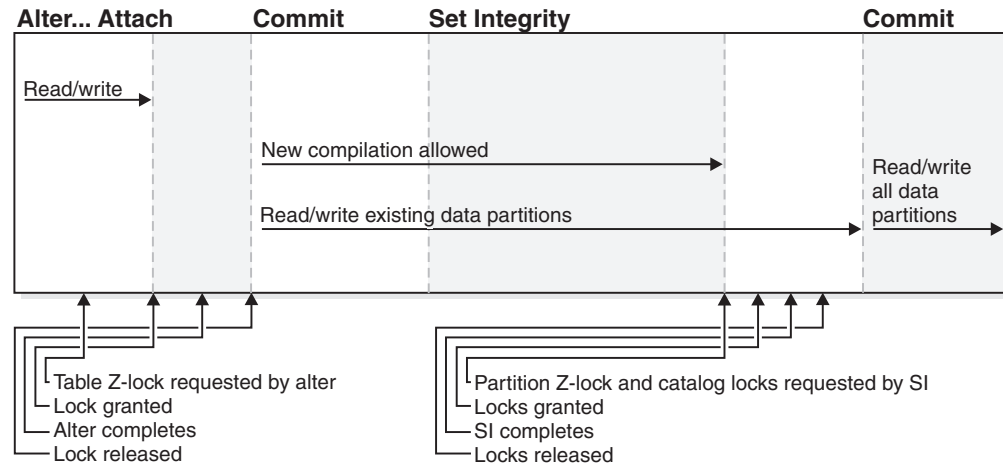


Figure 4. This figure demonstrates the stages of data availability during an ATTACH operation.

Note: While SET INTEGRITY is running, you cannot execute DDL or utility type operations on the table. The operations include but are not restricted to LOAD, REORG, REDISTRIBUTE, ALTER TABLE (for example, add columns, ADD, ATTACH, DETACH, TRUNCATE using ALTER to "not logged initially"), and INDEX CREATE.

```
SET INTEGRITY FOR stock ALLOW WRITE ACCESS
IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

Set integrity validates the data in the newly attached data partition.

Next, commit the transaction to make the table available for use.

```
COMMIT WORK;
```

Any rows that are out of range, or violate other constraints, are moved to the exception table stock_ex. You can query stock_ex to inspect the violating rows, and possibly to clean them up and re-insert them into the table.

Related concepts:

- "Data partitions" in *Administration Guide: Planning*
- "Asynchronous index cleanup" in *Performance Guide*
- "Attributes of detached data partitions" on page 354
- "Optimization strategies for partitioned tables" in *Performance Guide*
- "Partitioned materialized query table behavior" on page 206
- "Partitioned tables" in *Administration Guide: Planning*

Related tasks:

- "Adding data partitions to partitioned tables" on page 356
- "Altering a table" on page 297
- "Altering partitioned tables" on page 336
- "Approaches to defining ranges on partitioned tables" on page 195

- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352
- “Dropping a data partition” on page 358
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “LOAD command” in *Command Reference*
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “SET INTEGRITY statement” in *SQL Reference, Volume 2*
- “Command Line Processor (CLP) samples” in *Samples Topics*

Attaching a data partition

Table partitioning allows for the efficient roll-in and roll-out of table data. This efficiency is achieved by using the ATTACH PARTITION and DETACH PARTITION clauses of the ALTER TABLE statement. Rolling in partitioned table data allows you to easily incorporate a new range into a partitioned table as an additional data partition.

The ATTACH PARTITION clause takes an existing table (source table) and attaches it as a new data partition to the target table. The newly attached data partition is initially inaccessible to queries. The remainder of the table remains online. A call to the SET INTEGRITY statement is required to bring the attached data partition online.

Prerequisites:

To alter a table to attach a data partition, the privileges held by the authorization ID of the statement must include at least one of the following authorities and privileges on the source table:

- SELECT privilege on the source table and DROPIN privilege on the schema of the source table
- CONTROL privilege on the source table
- SYSADM or DBADM authority.

Restrictions and usage guidelines:

The following conditions must be met before you can attach a data partition:

- The table to which you want to attach the new data partition (that is, the target table) must be an existing partitioned table.
- The source table must be an existing non-partitioned table or a partitioned table with only a single data partition, and with no ATTACHED or DETACHED data partitions. To attach multiple data partitions, it is necessary to issue multiple ATTACH statements.
- The source table cannot be hierarchical (typed table).
- The source table cannot be a range-clustered table (RCT).
- The table definition for a source table must match the target table.
- The number, type, and ordering of columns must match for the source and target tables.

- For both tables, columns must match in terms of whether they contain default values. If the source column is created by using the ALTER TABLE ADD COLUMN, that is, SYSCOLUMNS.ADD_DEFAULT = 'Y', the existDefault value (SYSCOLUMNS.ADDED_DEFAULT) must match that of the target column.
- For both tables, columns must match in terms of whether they allow NULL or not.
- The Compression clause, including both VALUE COMPRESSION and SYSTEM COMPRESSION DEFAULT values must match for the source and target tables.
- Use of the APPEND clause with data capture option, and the not logged initially option must match.
- Attaching a data partition is allowed even when the target column is a generated column and the source column is not a generated column. This statement SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FORCE GENERATED generates the values for the generated column of the attached rows. The column matching a generated column must match in type and nullability. There are no required default values for this column. The recommended approach is to guarantee that the source table of the ATTACH has the correct generated value in the generated column. Then, you are not required to use the FORCE GENERATED option. The following statement can be used:


```
SET INTEGRITY FOR T GENERATED COLUMN IMMEDIATE UNCHECKED
(indicates to the system to bypass checking of generated column)
SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR EXCEPTION
IN T USE T_EX (performs integrity checking of the attached partition but
will not check for generated column correctness)
```
- Attaching a data partition is allowed even when the target column is identity and the source column is non-identity. The statement SET INTEGRITY IMMEDIATE CHECKED does not generate identity values for the attached rows. The statement SET INTEGRITY FOR T GENERATE IDENTITY ALLOW WRITE ACCESS IMMEDIATE CHECKED fills in the identity values for the attached rows. The column matching an identity column must match in type and nullability. There is no requirement on the default values of this column. The recommended approach is for you to fill in the correct identity values at the staging table. Then after the ATTACH, there is no requirement to use the GENERATE IDENTITY option because the identity values are already guaranteed in the source table.
- For tables whose data is distributed across database partitions, the source table must also be distributed, in the same database partition group using the same distribution key and the same distribution map.
- The source table must be droppable (that is, it cannot have RESTRICT DROP set).
- If a DATAPARTITIONNAME is specified, it must not already exist in the target table.
- If the target table is an multidimensional clustering (MDC) table, the source table must also be an MDC table.
- The data table space for the source table must match the data table spaces for the target table in type (that is, DMS or SMS), pageSize, extentSize and database partition group. A warning is returned to the user if the prefetchSize do not match. The long table space for the source table must match the long table spaces for the target table in type, database partition group and pageSize.

Procedure:

You can alter a table from the DB2 Control Center or the DB2 command line processor (CLP).

To use the DB2 Control Center to alter a partitioned table and to attach a data partition to the table:

1. Expand the Table folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click on the table that you want to modify and select Open Data Partitions from the list of options.
3. In the Open Data Partitions window click the Attach button.
4. In the Attach Data Partition window specify the name, and boundary specifications of the data partition to attach.
5. In the Open Data Partitions window click OK to modify the table.

To use the DB2 command line to alter a partitioned table and to attach a data partition to the table, issue the **ALTER TABLE** statement

Related concepts:

- “Data partitions” in *Administration Guide: Planning*
- “Partitioned tables” in *Administration Guide: Planning*
- “Attributes of detached data partitions” on page 354
- “Resolving a mismatch when trying to attach a data partition to a partitioned table” on page 348

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering a table” on page 297
- “Altering partitioned tables” on page 336
- “Creating a new source table using db2look” on page 210
- “Detaching a data partition” on page 352
- “Dropping a data partition” on page 358
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “SYSCAT.COLUMNS catalog view” in *SQL Reference, Volume 1*

Resolving a mismatch when trying to attach a data partition to a partitioned table

The following section provides guidelines for correcting various types of mismatches that can occur when attempting to attach a data partition to a partitioned table when issuing the ALTER TABLE ...ATTACH PARTITION statement. You can achieve agreement between tables by modifying the source table to match the characteristics of the target table, or by modifying the target table to match the characteristics of the source table. The source table is the existing table you want to attach to a target table. The target table is the table to which you want to attach the new data partition.

One suggested approach to performing a successful attach is to use the exact CREATE TABLE statement for the source table as you did for the target table, but without the PARTITION BY clause. In cases where it is difficult to modify the characteristics of either the source or target tables for compatibility, you can create a new source table that is compatible with the target table. For details on creating a new source see, *Creating a new source table using db2look* *Creating a new source table using db2look*.

To help you prevent a mismatch from occurring, refer to the Restrictions and usage guidelines section of *Attaching a data partition*. The section outlines conditions that must be met before you can successfully attach a data partition. Failure to meet the listed conditions returns error SQL20408N or SQL20307N.

The following sections describe the various types of mismatches that can occur and provides the suggested steps to achieve agreement between tables:

The (value) compression clause (the COMPRESSION column of SYSCAT.TABLES) does not match. (SQL20307N reason code 2):

To achieve value compression agreement, use one of the following statements:

```
ALTER TABLE... ACTIVATE VALUE COMPRESSION  
or  
ALTER TABLE... DEACTIVATE VALUE COMPRESSION
```

To achieve row compression agreement use one of the following statements:

```
ALTER TABLE... COMPRESS YES  
or  
ALTER TABLE... COMPRESS NO
```

The APPEND mode of the tables does not match. (SQL20307N reason code 3):

To achieve append mode agreement use one of the following statements:

```
ALTER TABLE ... APPEND ON  
or  
ALTER TABLE ... APPEND OFF
```

The code pages of the source and target table do not match. (SQL20307N reason code 4):

Create a new source

The source table is a partitioned table with more than one data partition or with attached or detached data partitions. (SQL20307N reason code 5):

Detach data partitions from the source table until there is a single visible data partition using the statement:

```
ALTER TABLE ... DETACH PARTITION
```

Include any necessary SET INTEGRITY statements. If the source table has indexes, you might not be able to attach the source table immediately. Detached data partitions remain detached until all indexes are cleaned-up of detached keys. If you want to perform an attach immediately, drop the index on the source table. Otherwise, create a new source.

The source table is a system table, a view, a typed table, a table ORGANIZED BY KEY SEQUENCE or a declared global temporary table. (SQL20307N reason code 6):

Create a new source.

The target and source table are the same. (SQL20307N reason code 7):

You cannot attach a table to itself. Determine the correct table to use as the source or target table.

The NOT LOGGED INITIALLY clause was specified for either the source table or the target table, but not for both. (SQL20307N reason code 8):

Either make the table that is not logged initially be logged by issuing the COMMIT statement, or make the table that is logged be not logged initially by entering the statement:

```
ALTER TABLE .... ACTIVATE NOT LOGGED INITIALLY
```

The DATA CAPTURE CHANGES clause was specified for either the source table or the target table, but not both. (SQL20307N reason code 9):

To enable data capture changes on the table that does not have data capture changes turned on, run the following statement:

```
ALTER TABLE ... DATA CAPTURE CHANGES
```

To disable data capture changes on the table that does have data capture changes turned on, run the statement:

```
ALTER TABLE ... DATA CAPTURE NONE
```

The distribution clauses of the tables do not match. The distribution key must be the same for the source table and the target table. (SQL20307N reason code 10):

It is recommended that you create a new source table. You cannot change the distribution key of a table spanning multiple database partitions. To change a distribution key on tables in single-partition database, run the following statements:

```
ALTER TABLE ... DROP DISTRIBUTION;  
ALTER TABLE ... ADD DISTRIBUTION(key-specification)
```

Only one of the tables has an ORGANIZE BY DIMENSIONS clause specified or the organizing dimensions are different. (SQL20307N reason code 11):

Create a new source.

The data type of the columns (TYPENAME) does not match. (SQL20408N reason code 1):

To correct a mismatch in data type, issue the statement:

```
ALTER TABLE ... ALTER COLUMN ... SET DATA TYPE...
```

The nullability of the columns (NULLS) does not match. (SQL20408N reason code 2):

To alter the nullability of the column that does not match for one of the tables issue one of the following statements:

```
ALTER TABLE... ALTER COLUMN... DROP NOT NULL  
or  
ALTER TABLE... ALTER COLUMN... SET NOT NULL
```

The implicit default value (SYSCAT.COLUMNS IMPLICITVALUE) of the columns are incompatible. (SQL20408N reason code 3):

Create a new source table. Implicit defaults must match exactly if both the target table column and source table column have implicit defaults (if IMPLICITVALUE is not NULL).

If IMPLICITVALUE is not NULL for a column in the target table and IMPLICITVALUE is not NULL for the corresponding column in the source table, each column was added after the original CREATE TABLE statement for the table. In this case, the value stored in IMPLICITVALUE must match for this column.

There is a situation, where through migration from a pre-V9.1 table or through attach of a data partition from a pre-V9.1 table, that IMPLICITVALUE is not NULL because the system did not know whether or not the column was added after the original CREATE TABLE statement. If the database is not certain whether the column is added or not, it is treated as added. An added column is a column created as the result of an ALTER TABLE ...ADD COLUMN statement. In this case, the statement is not allowed because the value of the column could become corrupted if the attach were allowed to proceed. You must copy the data from the source table to a new table (with IMPLICITVALUE for this column NULL) and use the new table as the source table for the attach operation.

The code page (COMPOSITE_CODEPAGE) of the columns does not match. (SQL20408N reason code 4):

Create a new source table.

The system compression default clause (COMPRESS) does not match. (SQL20408N reason code 5):

To alter the system compression of the column issue one of the following statements to correct the mismatch:

```
ALTER TABLE ... ALTER COLUMN ... COMPRESS SYSTEM DEFAULT  
or  
ALTER TABLE ... ALTER COLUMN ... COMPRESS OFF
```

Related concepts:

- “Data partitions” in *Administration Guide: Planning*
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Creating partitioned tables” on page 193
- “Approaches to migrating existing tables and views to partitioned tables” on page 198
- “Attaching a data partition” on page 346

- “Detaching a data partition” on page 352
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “SYSCAT.COLUMNS catalog view” in *SQL Reference, Volume 1*

Detaching a data partition

Table partitioning allows for the efficient roll-in and roll-out of table data. This efficiency is achieved by using the ATTACH PARTITION and DETACH PARTITION clauses of the ALTER TABLE statement.

Rolling-out partitioned table data allows you to easily separate ranges of data from a partitioned table. Once a data partition is detached into a separate table, the table can be handled in several ways. You can drop the separate table (whereby, the data from the data partition is destroyed); archive it or otherwise use it as a separate table; attach it to another partitioned table such as a history table; or you can manipulate, cleanse, transform and reattach to the original or some other partitioned table.

If the source table is a multidimensional clustered table (MDC), access to the newly detached table is not allowed in the same unit of work as the ALTER TABLE ...DETACH operation. Block indexes are created upon first access to the table after the ALTER TABLE ...DETACH operation is committed. Access time is reduced while the block indexes are created.

Prerequisites:

To detach a data partition from a partitioned table you must have the following authorities or privileges:

- The user performing the DETACH operation must have the authority needed to ALTER, to SELECT from and to DELETE from the source table.
- The user must also have the authority needed to create the target table. Therefore, to alter a table to detach a data partition, the privilege held by the authorization ID of the statement must include at least one of the following authorities or privileges on the target table:
 - SYSADM or DBADM authority
 - CREATETAB authority on the database and USE privilege on the table spaces used by the table as well as one of:
 - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema.

Note: When detaching a data partition, the authorization ID of the statement is going to effectively perform a CREATE TABLE statement and therefore must have the necessary privileges to perform that operation. The authorization ID of the ALTER TABLE statement becomes the definer of the new table with CONTROL authority, as if the user had issued the CREATE TABLE

statement. No privileges from the table being altered are transferred to the new table. Only the authorization ID of the ALTER TABLE statement and DBADM or SYSADM have access to the data immediately after the ALTER TABLE ...DETACH PARTITION statement.

Restrictions:

You must meet the following conditions before you can perform a DETACH operation:

- The table to be detached from (source table) must exist and be a partitioned table.
- The data partition to be detached must exist in the source table.
- The source table must have more than one data partition. A partitioned table must have at least one data partition. Only visible and attached data partitions pertain in this context. An attached data partition is a data partition that is attached but not yet validated by the SET INTEGRITY statement.
- The name of the table to be created by the DETACH operation (target table) must not exist.
- DETACH is not allowed on a table that is the parent of an enforced referential integrity (RI) relationship.
- If there are any dependent tables that need to be incrementally maintained with respect to the detached data partition (these dependents table are referred to as detached dependent tables), then the newly detached table is initially inaccessible. The table will be marked with an L in the TYPE column of the SYSCAT.TABLES catalog view. This is referred to as a detached table. This prevents the table from being read, modified or dropped until the SET INTEGRITY statement is run to incrementally maintain the detached dependent tables. After the SET INTEGRITY statement is run on all detached dependent tables, the detached table is transitioned to a regular table where it becomes fully accessible.

Procedure:

You can alter a table from the DB2 Control Center or the DB2 command line processor.

To use the DB2 Control Center to alter a partitioned table and to detach a data partition from the table:

1. Expand the **Table** folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click the table that you want to modify and select **Open Data Partitions** from the list of options.
3. In the Open Data Partitions window, select a data partition to detach.
4. Click the **Detach** button.
5. In the Detach Data Partition window, specify the table (schema and name) to create upon detach.
6. In the Open Data Partitions window, click **OK** to modify the table.

To use the DB2 command line to alter a partitioned table and to detach a data partition from the table, issue the **ALTER TABLE** statement with the DETACH PARTITION clause.

Related concepts:

- “Attributes of detached data partitions” on page 354
- “Data partitions” in *Administration Guide: Planning*

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Altering a table” on page 297
- “Dropping a data partition” on page 358
- “Attaching a data partition” on page 346
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338

Attributes of detached data partitions

When you detach a data partition from a partitioned table using the DETACH PARTITION clause of the ALTER TABLE statement, it becomes a stand-alone target table. Many attributes of the new target table are inherited from the source table. Any attributes not inherited from the source table are set as if the user executing the DETACH operation is creating the target table.

Note: If there are detached dependents then the detached data partition does not become a stand-alone table at detach time. In this case, the SET INTEGRITY statement must be issued to complete the detach and make the table accessible.

Attributes inherited by the target table:

Attributes inherited by the target table include:

- The following column definitions:
 - Column name
 - Data type (includes length and precision for types that have length and precision, such as CHAR and DECIMAL)
 - NULLability
 - Column default values
 - Code page (CODEPAGE column of SYSCAT.COLUMNS catalog view)
 - Logging for LOBs (LOGGED column of SYSCAT.COLUMNS catalog view)
 - Compaction for LOBs (COMPACT column of SYSCAT.COLUMNS catalog view)
 - Compression (COMPRESS column of SYSCAT.COLUMNS catalog view)
 - Type of hidden column (HIDDEN column of SYSCAT.COLUMNS catalog view)
 - Column order

- If the source table is a multidimensional clustering table (MDC), the target table is also an MDC table, defined with the same dimension columns. Access to the newly detached table is not allowed in the same unit of work as the detach when the source table is MDC.
- Block index definitions. The indexes are rebuilt on first access to the newly detached independent table after the DETACH operation is committed.
- The table space id and table object id are inherited from the data partition, not from the source table. This is because no table data is moved during a DETACH operation. In catalog terms, the TBSPACEID column of the SYSCAT.DATAPARTITIONS catalog view from the source data partition becomes the TBSPACEID column of the SYSCAT.TABLES catalog view. When translated into a table space name, it is the TBSPACE column of SYSCAT.TABLES catalog view in the target table. The PARTITIONOBJECTID column of the SYSCAT.DATAPARTITIONS catalog view from the source data partition becomes the TABLEID column of the SYSCAT.TABLES catalog view in the target table.
- The LONG_TBSPACEID column of the SYSCAT.DATAPARTITIONS catalog view from the source data partition is translated into a table space name and becomes the LONG_TBSPACE column of SYSCAT.TABLES of the target table.
- Table space location
- ID of distribution map for a multi-partition database (PMAP_ID column of SYSCAT.TABLES catalog view)
- Percent free (PCTFREE column of SYSCAT.TABLES catalog view)
- Append mode (APPEND_MODE column of SYSCAT.TABLES catalog view)
- Preferred lock granularity (LOCKSIZE column of SYSCAT.TABLES catalog view)
- Data Capture (DATA_CAPTURE column of SYSCAT.TABLES catalog view)
- VOLATILE (VOLATILE column of SYSCAT.TABLES catalog view)
- DROPRULE (DROPRULE column of SYSCAT.TABLES catalog view)
- Compression (COMPRESSION column of SYSCAT.TABLES catalog view)
- Maximum free space search (MAXFREESPACESEARCH column of SYSCAT.TABLES catalog view)

Note: Partitioned hierarchical or temporary tables, range-clustered tables, and partitioned views are not supported.

Attributes not inherited from the source table:

Attributes not inherited from the source table include:

- The target table type is not inherited. The target table is always a regular table.
- Privileges and Authorities
- Schema
- Generated columns, identity columns, check constraints, referential constraints. In the case where a source column is a generated column or an identity column, the corresponding target column has no explicit default value, meaning it has a default value of NULL.
- Index table space (INDEX_TBSPACE column of the SYSCAT.TABLES catalog view). The value is set to NULL. Indexes for the table resulting from the DETACH will be in the same table space as the table.
- Triggers
- Primary key
- Statistics

- All other attributes not mentioned in the list of attributes explicitly inherited from the source table.

Related concepts:

- “Data partitions” in *Administration Guide: Planning*
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Altering partitioned tables” on page 336
- “Altering a table” on page 297
- “Dropping a data partition” on page 358
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352
- “Rotating data in a partitioned table” on page 339

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338
- “Examples of rolling in and rolling out partitioned table data” on page 342

Adding data partitions to partitioned tables

You can use the ALTER TABLE statement to modify a partitioned table after the table is created. Specifically, you can use the ADD PARTITION clause to add a new data partition to an existing partitioned table. Adding a data partition to a partitioned table is more appropriate than attaching a data partition in situations where data is added to the data partition over time, when data is trickling in rather than rolling in from an external source, or when you are inserting or loading data directly into a partitioned table. Specific examples include daily loads of data into a data partition for January data, or ongoing inserts of individual rows.

Restrictions and usage guidelines:

- You cannot add a data partition to a non-partitioned table. For details on migrating an existing table to a partitioned table, see Approaches to migrating existing tables and views to partitioned tables
- The range of values for each new data partition, are determined by the STARTING and ENDING clauses.
- One or both of the STARTING and ENDING clauses must be supplied.
- The new range must not overlap with the range of an existing data partition.
- When adding a new data partition before the first existing data partition, the STARTING clause must be specified. Use MINVALUE to make this range open ended.
- Likewise, the ENDING clause must be specified if you want to add a new data partition after the last existing data partition. Use MAXVALUE to make this range open ended.
- If the STARTING clause is omitted, then the database manufactures a starting bound just after the ending bound of the previous data partition. Likewise, if the ENDING clause is omitted, the database manufactures an ending bound just before the starting bound of the next data partition.

- The start-clause and end-clause syntax is the same as specified in the CREATE TABLE statement.
- If no IN or LONG IN clause is specified for ADD PARTITION, the table space in which to place the data partition is chosen using the same method as is used by the CREATE TABLE statement.
- Packages are invalidated during the ALTER TABLE ...ADD PARTITION operation.
- The newly added data partition is available once the ALTER TABLE statement is committed.

Omitting the STARTING or ENDING bound for an ADD or ATTACH operation is also used to fill a gap in range values. Here is an example of filling in a gap using the ADD operation where only the starting bound is specified:

```
CREATE TABLE hole (c1 int) PARTITION BY RANGE (c1)
(STARTING FROM 1 ENDING AT 10, STARTING FROM 20 ENDING AT 30);
DB20000I The SQL command completed successfully.
ALTER TABLE hole ADD PARTITION STARTING 15;
DB20000I The SQL command completed successfully.
SELECT SUBSTR(tabname, 1,12) tabname,
SUBSTR(datapartitionname, 1, 12) datapartitionname,
seqno, SUBSTR(lowvalue, 1, 4) lowvalue, SUBSTR(highvalue, 1, 4) highvalue
FROM SYSCAT.DATAPARTITIONS WHERE TABNAME='HOLE' ORDER BY seqno;
TABNAME DATAPARTITIONNAME SEQNO LOWVALUE HIGHVALUE
-----
HOLE PART0 0 1 10
HOLE PART2 1 15 20
HOLE PART1 2 20 30
```

3 record(s) selected.

Procedure:

You can alter a table from the DB2 Control Center or the DB2 command line processor (CLP).

To use the DB2 Control Center to alter a partitioned table and to add a new data partition to the table:

1. Expand the Table folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click the table that you want to modify and select Open Data Partitions from the list of options.
3. In the Open Data Partitions window, select a data partition to add.
4. Click the Add button.
5. In the Add Data Partition window, specify the name, boundary specifications and source table of the data partition.
6. In the Open Data Partitions window, click OK to modify the table.

To use the DB2 command line to alter a partitioned table and to add a new data partition to the table, issue the **ALTER TABLE** statement with the ADD PARTITION clause.

Example 1: Add a data partition to an existing partitioned table holding a range of values 901 to 1000 inclusive. Assume table sales holds nine ranges 0-100, 101-200,

and so on, up to the value of 900. The example adds an additional range at the end of the table, indicated by the exclusion of the STARTING clause:

```
ALTER TABLE sales ADD PARTITION dp10
(ENDING AT 1000 INCLUSIVE)
```

Related concepts:

- “Attributes of detached data partitions” on page 354
- “Data partitions” in *Administration Guide: Planning*
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Approaches to migrating existing tables and views to partitioned tables” on page 198
- “Rotating data in a partitioned table” on page 339
- “Altering partitioned tables” on page 336
- “Dropping a data partition” on page 358
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338

Dropping a data partition

You can drop a data partition using the **ALTER TABLE** statement with the **DETACH PARTITION** clause followed by the **DROP TABLE** statement to drop the separate table.

Prerequisites:

To detach a data partition from a partitioned table the user must have the following authorities or privileges:

- The user performing the **DETACH** must have the authority needed to **ALTER**, to **SELECT** from and to **DELETE** from the source table.
- The user must also have the authority needed to **CREATE** the target table. Therefore, in order to alter a table to detach a data partition, the privilege held by the authorization ID of the statement must include at least one of the following on the target table:
 - **SYSADM** or **DBADM** authority
 - **CREATETAB** authority on the database and **USE** privilege on the table spaces used by the table as well as one of:
 - **IMPLICIT_SCHEMA** authority on the database, if the implicit or explicit schema name of the table does not exist
 - **CREATEIN** privilege on the schema, if the schema name of the table refers to an existing schema.

To drop a table the user must have the following authorities or privileges:

- You must either be the definer as recorded in the DEFINER column of SYSCAT.TABLES, or have at least one of the following privileges:
 - SYSADM or DBADM authority
 - DROPIN privilege on the schema for the table
 - CONTROL privilege on the table

Note: The implication of the detach data partition case is that the authorization ID of the statement is going to effectively issue a **CREATE TABLE** statement and therefore must have the necessary privileges to perform that operation. The table space is the one where the data partition that is being detached already resides. The authorization ID of the **ALTER TABLE** statement becomes the definer of the new table with CONTROL authority, as if the user had issued the **CREATE TABLE** statement. No privileges from the table being altered are transferred to the new table. Only the authorization ID of the **ALTER TABLE** statement and DBADM or SYSADM have access to the data immediately after the ALTER TABLE ... DETACH PARTITION operation.

Procedure:

You can detach a data partition of a partitioned table from the DB2 Control Center or from the DB2 command line processor (CLP).

To use the DB2 Control Center to detach a data partition of a partitioned table:

1. Expand the **Tables** folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click the table that contains the data partition you want to detach, and select **Open Data Partitions** from the list of actions.
3. In the Open Data Partitions window, click the **Detach** button.
4. Specify the required fields.

To use the DB2 command line to detach a data partition of a partitioned table, issue the ALTER TABLE statement with the DETACH PARTITION clause.

You can drop a table from the DB2 Control Center or from the DB2 command line processor (CLP).

To use the DB2 Control Center to drop a table:

1. Expand the Tables folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click on the table you want to drop, and select Drop from the pop-up menu.
3. Verify your change in the Confirmation window.

To use the DB2 command line to drop a table, issue the **DROP TABLE** statement.

Example:

In this example, the dec01 data partition is detached from table stock and placed in table junk. You can then drop table junk, effectively dropping the associated data partition.

```
ALTER TABLE stock DETACH PART dec01 INTO junk;  
DROP TABLE junk;
```

Note: To make the DETACH operation as fast as possible, index cleanup on the source table is done automatically using a background asynchronous index cleanup process. If there are detached dependents then the detached data partition does not become a stand-alone table at detach time. In this case, the SET INTEGRITY statement must be issued to complete the detach and make the table accessible.

Related concepts:

- “Attributes of detached data partitions” on page 354
- “Data partitions” in *Administration Guide: Planning*
- “Partitioned tables” in *Administration Guide: Planning*

Related tasks:

- “Adding data partitions to partitioned tables” on page 356
- “Altering partitioned tables” on page 336
- “Attaching a data partition” on page 346
- “Detaching a data partition” on page 352
- “Rotating data in a partitioned table” on page 339

Related reference:

- “Examples of rolling in and rolling out partitioned table data” on page 342
- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “Guidelines and restrictions on altering partitioned tables with attached or detached data partitions” on page 338

Updating table and view contents using the MERGE statement

The DB2 database manager 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 *SQL Reference, Volume 2*

Recovering inoperative summary tables

Summary tables can become *inoperative* as a result of a revoked SELECT privilege on an underlying table.

Procedure:

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:

- “GRANT (Table, View, or Nickname Privileges) statement” in *SQL Reference, Volume 2*
- “SYSCAT.VIEWS catalog view” in *SQL Reference, Volume 1*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*
- “DROP statement” in *SQL Reference, Volume 2*

Dropping or deleting tables

This section describes how to drop or delete tables.

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 *SQL Guide*

Related reference:

- “DELETE statement” in *SQL Reference, Volume 2*
- “UPDATE statement” in *SQL Reference, Volume 2*

Deleting the contents of staging tables

You can delete the contents of a staging table.

Prerequisites:

To delete the contents of a staging table, you need the following authorities:

- SYSADM or DBADM authority
- CONTROL privileges on the staging table being pruned

Procedure:

To delete the contents of a staging table using the Control Center:

1. Open the Set Integrity window: From the Control Center, expand the object tree until you find the **Tables** folder. Click on the **Tables** folder. Any existing tables are displayed in the pane on the right side of the window. Right-click the table you want and select **Set Integrity** from the pop-up menu. The Set Integrity window opens.
2. Review the **Current integrity status** of the table you are working with.
3. If the table is in Set Integrity Pending state, select the **Immediate and checked** and the **Prune** check box in the **Options** group box to delete the contents of the staging table and to propagate to the staging table.
4. If the table is not in Set Integrity Pending state, select the **Prune** radio button to delete the contents of the staging table without propagating to the staging table.
Note: If you select the **Immediate and checked** radio button, the table will be brought out of Set Integrity Pending state.

To delete the contents of a staging table using the command line, use the **SET INTEGRITY** statement.

Related tasks:

- “Checking for constraint violations using SET INTEGRITY” on page 230

Related reference:

- “SET INTEGRITY statement” in *SQL Reference, Volume 2*

Dropping a table

A table can be dropped with a DROP TABLE SQL statement.

When a table is dropped, the row in the SYSCAT.TABLES catalog that contains information about that table is dropped, and any other objects that depend on the table are affected. For example:

- All column names are dropped.
- Indexes created on any columns of the table are dropped.
- All views based on the table are marked inoperative.
- 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 and XQuery 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 might not be immediately available for other operations.

Procedure:

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 366

Related tasks:

- “Dropping a user-defined temporary table” on page 364
- “Recovering inoperative views” on page 331

Related reference:

- “DROP statement” in *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 212

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*
- “SET SCHEMA statement” in *SQL Reference, Volume 2*

Dropping a materialized query or staging table

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.

Procedure:

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 might be explicitly dropped with the DROP TABLE statement, or it might be dropped implicitly if any of the underlying tables are dropped.

A staging table might be explicitly dropped with the DROP TABLE statement, or it might be dropped implicitly when its associated materialized query table is dropped.

Related concepts:

- “Statement dependencies when changing objects” on page 366

Related tasks:

- “Creating a materialized query table” on page 201
- “Creating a staging table” on page 211

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*

Statement dependencies when changing objects

Statement dependencies include package and cached dynamic SQL and XQuery 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 and XQuery 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 query statement depends on an object and that object is dropped, the package or cached dynamic query 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 or XQuery 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 or XQuery statement might 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 rebound 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 rebound 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 recreation using the BIND command and an existing bind file" in *Developing Embedded SQL Applications*
- "Rebinding existing packages with the REBIND command" in *Developing Embedded SQL Applications*

Related reference:

- "SYSCAT.PACKAGEAUTH catalog view" in *SQL Reference, Volume 1*
- "SYSCAT.PACKAGEDEP catalog view" in *SQL Reference, Volume 1*
- "SYSCAT.PACKAGES catalog view" in *SQL Reference, Volume 1*
- "BIND command" in *Command Reference*
- "REBIND command" in *Command Reference*
- "DROP statement" in *SQL Reference, Volume 2*

Chapter 7. Using the DB2 administration tools

This section describes how to use the DB2 graphical user interface tools and includes some tasks that can only be performed using the graphical user interface. This section also discusses 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.

Starting the server DB2 administration tools

Use the Control Center to perform administrative tasks on systems, databases, and database objects. Objects are displayed in the Control Center on the object tree and in the contents pane .

To open the Control Center:

- In Windows, click **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**.
- In Linux, open the **IBM DB2** folder on the desktop and click **Control Center**.

Display the object in the Control Center.

Note: Some administration tasks are invoked from folders on the object tree, while others are invoked from individual object icons. For example, the task of creating a database is invoked from the **Databases** folder, while configuring a database is invoked on the database itself.

Right-click the object. A pop-up menu of all of the available administration actions for that object opens. Click the task in the pop-up menu. A window or notebook opens to guide you through the steps required to complete the action for the selected object.

For more information, see the *Administration Guide* , the *Command Reference* , and the *SQL Reference*.

Related concepts:

- “Control Center overview” on page 376

Shutting down server DB2 administration tools

Use the **Shut Down DB2 Tools** menu choice to shut down all of the server DB2 administration tools.

When you shut down the server DB2 administration tools, all connections are dropped and the windows for all open centers close.

Related tasks:

- “Setting startup and default options for the DB2 administration tools” on page 436
- “Setting the server administration tools startup property” on page 434

Finding service level information about the DB2 administration tools environment

Use the About DB2 Administration Tools Environment window, About System window, or About Instance window to gather service level information about the DB2 administration tools. This information is used by the DB2 database service analyst to determine the exact service level of the DB2 administration tools running on your system.

To open the About DB2 Administration Tools Environment, click **Help**→**About**. To open the About System window or the About Instance window, select a system or instance and click **About** from the pop-up menu.

These windows contain the following information about the DB2 administration tools:

- **Product identifier** : identifies the product in the format *pppvrrm* , where *ppp* is the product, *vv* is the version, *rr* is the release, and *m* is the modification level.
- **Level identifier** , **Level** , **Build level** , and **PTF** : identifies the level of service applied to the DB2 administration tools. This information changes as FixPaks and other service items are applied.
- **Level of the Java code base** : only on the About DB2 Administration Tools Environment window.
- **Operating system** : only on the About System window.

To copy the information in the window to the clipboard, click **Copy to Clipboard**. You can then paste the information into a file, e-mail, or other application.

Related concepts:

- “Control Center overview” on page 376

Using the DB2 database help

Use the DB2 database help to find information about operating the centers and components of DB2 database system.

To find a topic when you are not sure where to start, use one or more of the following methods:

- Select a topic from the DB2 Information Center.
- Type the relevant keywords in the **Search** field of the DB2 Information Center, and click **GO** .
- Select a topic from the table of contents in the left pane of the help window.
- Look up a term in the glossary by clicking the **Glossary** link in the left pane help window.
- Look up a keyword in the online index by clicking the **Index** link in the left pane help window.

To get help for the current window, notebook, or wizard:

- Click the **Help** push button, if available.
- Windows, notebooks, and wizards have control-specific help known as infopops. You can display the help for a field or control by selecting it and pressing F1. If

the **Automatically display infopops** check box on the Help System page of the Tool Settings notebook is selected, you can also see the infopop by holding the mouse pointer over the field or control.

- Click on the hypertext links (underlined text), if any, that appear in the descriptions on the windows, notebooks, or wizard pages.



Help prefixed with the version icon, , indicates that it applies to a specific version of the product.

Printing a help topic:

To print a help topic, click the **File->Print** , or right-click anywhere in the topic text and click **Print** .

Help that indicates partitioned or single-partition database environments:

In the DB2 database help, graphics are used to indicate when information applies to a subset of situations:

-  Information that pertains only to partitioned database environments is prefixed with the icon shown at the beginning of this sentence.
-  Information that pertains only to single-partition database environments is prefixed with the icon shown at the beginning of this sentence.

Related tasks:

- “Setting up access to DB2 contextual help and documentation” on page 435

Related reference:

- “DB2 Help menu” on page 375

Environment-specific information

 Information marked with this icon pertains only to single-partition database environments.

 Information marked with this icon pertains only to partitioned database environments.

Related concepts:

- “Control Center overview” on page 376

Menus and toolbars

This section describes the menus and toolbars found in the DB2 administration tools.

DB2 toolbar



Use the toolbar icons to open DB2 tools, view the legend for DB2 objects, and view DB2 information. To get help on the toolbar icons, hold your cursor over each icon to display the hover help. Note that the number of icons on the toolbar varies depending on what centers are installed.

The toolbar icons are:



Control Center

Opens the Control Center to enable you to display all of your systems, databases, and database objects and perform administration tasks on them.



Replication Center

Opens the Replication Center to enable you to design your replication environment and set up your replication environment.



Satellite Administration Center

Opens the Satellite Administration Center to enable you to set up and administer satellites and the information that is maintained in the satellite control tables.



Command Editor

Opens the Command Editor to enable you to work with database commands, their results, and access plans for queries.



Task Center

Opens the Task Center to enable you to create, schedule, and execute tasks.



Health Center

Opens the Health Center to enable you to work with alerts generated while using the DB2 database manager.



Journal

Opens the Journal to enable you to schedule jobs that are to run unattended and view notification log entries.



License Center

Opens the License Center to enable you to display license status and usage information for the DB2 products installed on your system and use the License Center to configure your system for license monitoring.



Configuration Assistant

Opens the Configuration Assistant to enable you to configure your workstation to access your DB2 subsystem.



Tools Settings

Opens the Tools Settings notebook to enable you to customize settings and properties for the administration tools and for replication tasks.



Legend

Opens the Legend window that displays all of the object icons available in the Control Center by icon and name.



Help

Displays information about how to use help for this product.

Related tasks:

- “Changing the fonts for menus and text” on page 437

Related reference:

- “DB2 Help menu” on page 375
- “DB2 secondary toolbar” on page 373
- “DB2 Tools menu” on page 374

DB2 secondary toolbar



Use the toolbar below the contents pane to tailor the view of objects and information in the contents pane to suit your needs.



Sort

Opens the Sort window so that you can select the order in which objects are displayed in the contents pane. You can sort on any column, or on multiple columns, in the contents pane (ascending or descending).



Filter

Opens the Filter window so that you can filter the objects that appear in the contents pane.



Customize columns

Opens the Customize Columns window so that you can select the order of the informational columns in the contents pane and reorder, include, or exclude them.



Find

Opens the Find window so that you can search for a string in the columns of the contents pane.



Select all

Selects all of the objects in the contents pane.



Deselect all

Deselects all selected objects in the contents pane.



Expand all

Expands all of the objects in the contents pane.



Collapse all

Collapses all objects in the contents pane.

Default View

Click to display the default or named views. If you are using the default view, the button name is Default View. If you are using one of the named views, the button name reflects the name of the view that you are using.

View Click to display some of the View menu options.

Related tasks:

- “Changing the fonts for menus and text” on page 437

Related reference:

- “DB2 Help menu” on page 375
- “DB2 toolbar” on page 371
- “DB2 Tools menu” on page 374

DB2 Tools menu

Use the **Tools** menu to open any of the DB2 Administration tools. Some of the functions in this menu are also available by clicking the icons in the DB2 toolbar.

From this menu, you can select the following menu items. Depending on which tool you are using, some of these menu options will not display.

Wizards

Opens the Wizards window so that you have quick access to the more common DB2 wizards.

Control Center

Opens the Control Center to enable you to manage systems, DB2 database instances, DB2 Universal Database for OS/390 and z/OS subsystems, databases, and database objects such as tables and views. In the Control Center, you can display all of your systems, databases, and database objects and perform administration tasks on them.

Replication Center

Opens the Replication Center to enable you to administer relational data between DB2 servers or databases.

Satellite Administration Center

Opens the Satellite Administration Center so that you can set up and administer both satellites, and the information that is maintained in the satellite control tables at a central DB2 control server.

Command Editor

Opens the Command Editor to enable you to execute DB2 CLP commands and query statements, z/OS or OS/390 operating system commands, or command scripts. This editor also lets you view a graphical representation of the access plan for explained SQL and XQuery statements.

Task Center

Opens the Task Center to enable you to create, schedule, and run tasks such as DB2 or operating system command scripts, MVS™ shell scripts, JCL scripts.

Health Center

Opens the Health Center to enable you to monitor instances using the Health Center. This center also alerts you to potential problems and

provides recommendations to resolve those problems. You can also use specific monitoring tools, such as the Memory Visualizer to drill-down into specific performance areas.

Journal

Opens the Journal to enable you to view historical information generated within the Control Center and its components.

License Center

Opens the License Center to enable you to display license status and usage information for the DB2 products installed on your system and use the License Center to configure your system for license monitoring.

Customize Control Center

Opens the Control Center View window where you can specify the Control Center view that you want to display: Basic, Advanced, or Custom.

Tools Settings

Opens the Tools Settings notebook so that you can customize settings and set properties for the administration tools, for replication tasks, for Health Center notification, and for default scheduling schemes.

Related tasks:

- “Changing the fonts for menus and text” on page 437

Related reference:

- “DB2 Help menu” on page 375
- “DB2 secondary toolbar” on page 373
- “DB2 toolbar” on page 371

DB2 Help menu

Use the Help menu on DB2 Administration tools to display online help and information about the DB2 tool you are using.

Help index

Opens the DB2 master index.

General Help

Displays getting started help for the center or component you are working with.

Keyboard help

Displays information about how to use accelerator keys for this product.

Using help

Displays information about how to use online help for this product.

DB2 Tutorials

Opens a Web page that lists the tutorials available with this version of the DB2 product. The tutorials are available both in HTML and PDF format.

Product information

Displays information about the product.

About

Displays service information about the DB2 Administration tools.

Related concepts:

- “Control Center overview” on page 376

Related tasks:

- “Changing the fonts for menus and text” on page 437

Related reference:

- “DB2 secondary toolbar” on page 373
- “DB2 toolbar” on page 371
- “DB2 Tools menu” on page 374


Control Center

This section describes how to use the Control Center, including how it can be extended.

Control Center overview

Use the Control Center to manage and administer systems, DB2 database instances, DB2 Universal Database for z/OS subsystems, databases, and database objects such as tables and views. From the Control Center, you can also open other centers and tools to help you optimize queries, jobs, and scripts, perform data warehousing tasks, create stored procedures, and work with DB2 and IMS commands.

The Control Center supports the native XML data type for many of its functions. This allows database administrators to work with XML documents stored in XML columns alongside relational data.

Note: As you work with the Control Center, you might encounter information prefixed with the  icon. This means that the associated information applies only if you are working in a partitioned database environment.

To open the Control Center:

- In Windows, click **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**.
- In Linux, open the **IBM DB2** folder on the desktop and click **Control Center**.

Tasks from the Control Center

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

- Add DB2 database systems, federated systems, DB2 UDB for z/OS and OS/390 systems, IMSplexes, instances, databases, and database objects to the object tree.
- Manage database objects. You can create, alter, and drop databases, table spaces, tables, views, indexes, triggers, and schemas. You can also manage users.
- Manage data. You can load, import, export, and reorganize data. You can also gather statistics.
- Perform preventive maintenance by backing up and restoring databases or table spaces.
- Configure and tune instances and databases.
- Manage database connections, such as DB2 Connect servers and subsystems.
- Manage IMS systems.
- Manage DB2 UDB for z/OS and OS/390 subsystems.

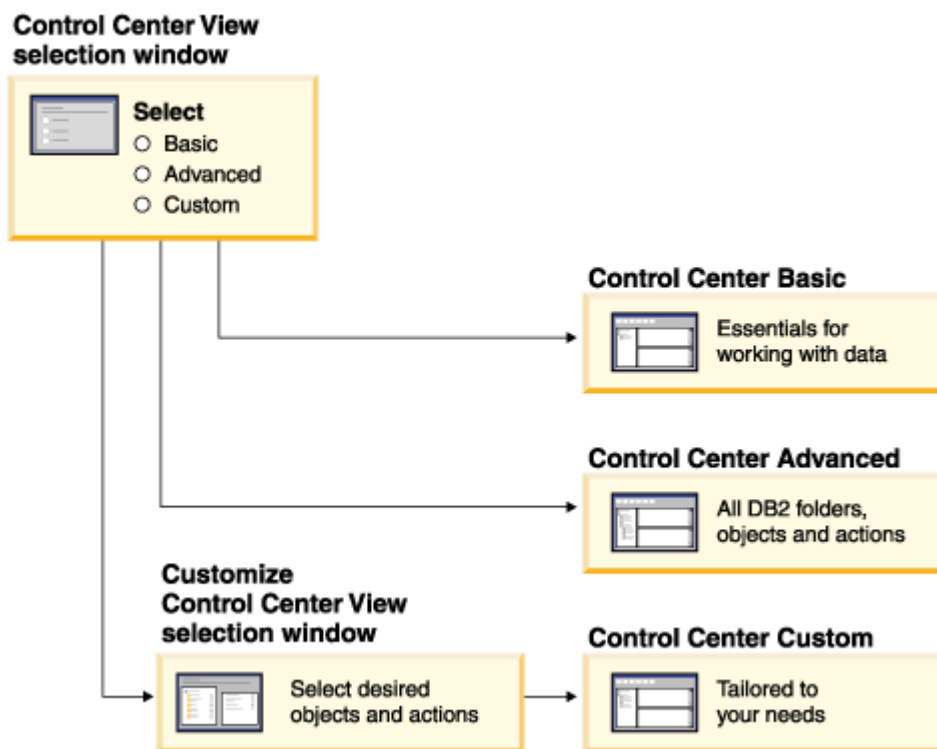
- Manage applications.
- Analyze queries using Visual Explain to look at access plans.
- Launch other tools such as the Command Editor and the Health Center.

In many cases, advisors, launchpads, and wizards are available to help you perform these tasks quickly and easily.

The Control Center interface

The Control Center interface is available in three different views:

- **Basic.** This view provides core DB2 database functionality, which includes the essential objects, such as databases, tables, and stored procedures.
- **Advanced.** This view displays all objects and actions available in the Control Center. This is the view that you should select if you are working in an enterprise environment and 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 to your specific needs.

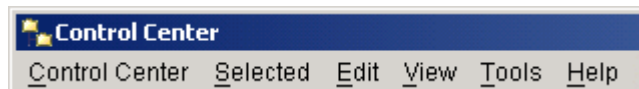


You can select or change your view by choosing **Tools** from the menu bar and selecting **Customize the Control Center**. You can then use your Control Center view to work with the various folders and the objects that they contain (the objects within a folder are called folder objects).

Working in the Control Center

The Control Center has six action areas that you can use to define, manage, and work with DB2 objects.

Menu bar



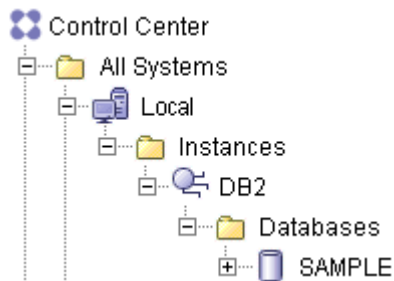
Use the menu bar to work with folders and folder objects in the Control Center, open other DB2 centers and tools, access advisors, wizards and launchpads, and display online help.

Control Center toolbar



Use the toolbar icons below the menu bar to access other DB2 centers and tools and display online help. Note that the icons on this toolbar reflect the set of administration tools installed and might be different than those shown in the graphic above.

Object tree



Use the object tree to display and work with folders and folder objects. Selecting an item displays related objects, actions, and information in the contents pane and the object details pane. Right-clicking an item displays a pop-up menu listing all the actions that you can perform on that item.

Contents pane



Use the contents pane to display and work with folder objects. The contents pane displays those objects that make up the contents of the folder that is selected in the object tree. Selecting an object displays its associated actions and information in the object details pane.

Contents pane toolbar



Use the toolbar below the contents pane to tailor the view of objects and information in the contents pane to suit your needs. These functions are also available from **Edit** and **View** in the menu bar.

Object Details pane

Database - SAMPLE

Alias name	: SAMPLE
System	: ALBARON1
Type	: Local

Use the object details pane to display information on and work with the folder or folder object that you have selected in the object tree or contents pane. If the object details pane is not displayed, select **View** from the menu bar and select **Show Object Details pane**.

The object details pane is only available for Windows, Linux, and UNIX operating systems, and only when database objects are selected from the object tree.

Accessing custom controls with the keyboard:

- You can use the keyboard to access controls found on the graphical user interface. For more information, see [Keyboard shortcuts and accelerators](#).

Accessing other types of help:

- Infopops: An infopop is pop-up information that is displayed when your mouse pointer is over a field or control in a window or notebook, or when field or control has focus and you press F1.
- Hover help: Hover help is pop-up information that is displayed when your mouse pointer is over an element of the interface (for example, an icon).

Accessing help for icons in the product:

- See [Control Center icons](#) to familiarize yourself with the various graphical images used in the Control Center.

Setting DB2 administration tools preferences:

- See [Tool Settings overview](#) to tailor the appearance of the Control Center (for example, the text font) to your own personal preferences.

Related concepts:

- [“Command Editor overview”](#) in *Online DB2 Information Center*
- [“Configuration Assistant overview”](#) in *Online DB2 Information Center*
- [“Guidelines for Control Center plugin developers”](#) on page 395
- [“Introducing the plug-in architecture for the Control Center”](#) on page 395
- [“Journal overview”](#) on page 418
- [“Task Center overview”](#) on page 416
- [“Visual Explain overview”](#) on page 451
- [“Writing plugins as Control Center extensions”](#) on page 397

Related tasks:

- [“Displaying objects in the Control Center”](#) on page 392
- [“Expanding and collapsing the Control Center object tree”](#) on page 389
- [“Getting help in the Control Center”](#) on page 385
- [“Managing database partitions from the Control Center”](#) on page 282
- [“Obtaining Control Center diagnostic information”](#) on page 393

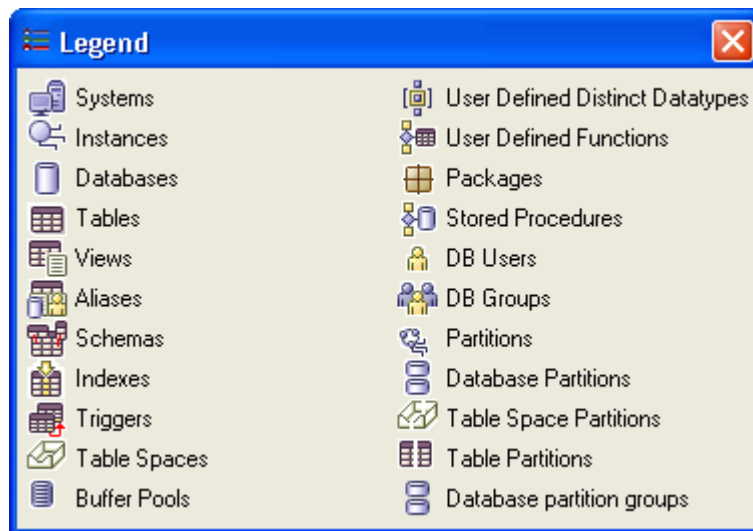
- “Opening new Control Centers” on page 382
- “Setting Command Editor options” on page 449
- “Setting up the DB2 administration server (DAS) to use the Configuration Assistant and the Control Center” on page 110

Related reference:

- “Control Center Legend” on page 380

Control Center Legend

Use the Legend window to see the object icons used in the Control Center and what they represent.



To open the Legend window, click  on the toolbar.

Objects:



System

A computer system defined to the DB2 system DB2 administration tools.



Instance

A database manager environment that is an image of the actual database manager environment. You can have several instances of a database manager on the same system.



Database

A DB2 relational database. A relational database presents data as a collection of tables.



Table

A named data object consisting of a specific number of columns and some unordered rows.



View

A logical table that consists of data that is generated by a query.

**Alias**

An alternative name used to identify a table, view, or database.

**Schema**

A collection of database objects such as tables, views, indexes, and triggers. It provides a logical classification of database objects.

**Index**

A set of pointers that are logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness on the rows in the table.

**Trigger**

An object in a database that is invoked indirectly by the database manager when a particular SQL statement is run.

**Table Space**

An abstraction of a collection of containers into which database objects are stored.

**Buffer Pool**

An area of storage in which all buffers of a program are kept.

**User-Defined Distinct Datatype**

A data type that is not native to the DB2 database manager and that was created by a user.

**User-Defined Function**

A function that is defined to the database management system and can be referenced in SQL queries.

**Package**

A control structure produced during program preparation that is used to execute SQL statements.

**Stored Procedures**

A block of procedural constructs and embedded SQL statements that is stored in a database and can be called by name.

**DB User**

A user that has specific database privileges.

**DB Group**

A group of users that has specific database privileges.

**Partition**

In a partitioned database environment, one part of either a database partition, a table space partition, or a portion of a table.

**Database Partition**

In a partitioned database environment, a part of the database that consists of its own user data, indexes, configuration files, and transaction logs.



Table Space Partition

In a partitioned database environment, table spaces that reside in database partition groups.



Table Partition

A table in a database partition group consisting of multiple partitions, some of its rows are stored in one partition, and other rows are stored in other partitions.



Database Partition Group

In a partitioned database environment, a named set of one or more database partitions.


A set of a particular object type is represented by a folder that has the icon for that type displayed on top of it. For example, a set of systems is represented by the following icon:



Related concepts:

- “Control Center overview” on page 376

Opening new Control Centers

To open a new Control Center, click the  icon on the toolbar, or right-click a system, subsystem, instance, database folder, or another object, and click **Open new Control Center** in the pop-up menu. The new Control Center opens in a separate window. If this action is performed in another center, the last Control Center opened is brought to the front.

The object tree in the new Control Center will start with the object you selected to open the new Control Center.

With a second Control Center, you can work with two or more objects that are not easily displayed in a single object tree or contents pane. This feature is especially useful when you want to look at the contents of two folders at the same time.

Related concepts:

- “Control Center overview” on page 376

Creating database objects

Use the Control Center to create database objects such as databases, tables, views, indexes, and triggers.

To create databases in the Control Center, you need either SYSADM or SYSCTRL authority. For information on the authorities needed to create other database objects, such as tables and table spaces, see the authorities and privileges information for creating the specific object.

To create a new database object, Open the Control Center:

- In Windows, click **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**.
- In Linux, open the **IBM DB2** folder on the desktop and click **Control Center**.

Expand the object tree to display a folder for the type of object that you want to create. Right-click the folder. A pop-up menu of all of the available actions for the object opens. Click the **Create** or **Create from Import** menu item, if it is available. A window or notebook opens to guide you through the process for creating the object.

Related concepts:

- “About databases” in *Administration Guide: Planning*
- “Control Center overview” on page 376

Changing system names displayed in the Control Center

Systems are displayed in the Control Center object tree, and each system represents a physical server. The system name that is displayed for each system is stored in the corresponding admin node directory entry for the system.

Note:

For remote systems, it is recommended that you use the remote hostname as the system name. This will ensure that the system names are unique in the Control Center object tree.

- When you change a system name displayed in the Control Center using the Change System window, the system name of the nodes under the same system are also updated.
- When you change a system name using the CLP, you will have to update each child node individually.

Prerequisites:

To change system names, you must have SYSADM or SYSCTRL authority.

Procedure:

To change a system name using either the Control Center or the Configuration Assistant:

1. Open the Change System window using one of the following methods:
 - From the Control Center, expand the object tree until you find the system that you want to change. Right-click the **All Systems** folder and select **Change** from the pop-up menu. The Change System window opens.
 - From the Configuration Assistant Advanced view, click the Systems tab. Select the system that you want to change and click **Selected->Change System**. The Change System window opens.

Note: You cannot open the Change System window from the Selected menu unless you are in the Advanced view. To switch to the Advanced view, select **View->Advanced View**.
2. Change the system name and other fields as required. If you are changing protocol information, you may require the assistance of your network or database administrator.

To change a system name using the CLP, use the **CATALOG ADMIN NODE** and **CATALOG NODE** commands.

For example, suppose that you have the following entries in the node directory and in the admin node directory:

```
DB2 LIST ADMIN NODE DIRECTORY SHOW DETAIL
```

```
Node Directory
Number of entries in the directory = 1
```

```
Node 1 entry:
```

```
Node name = HONCHO
Comment =
Directory entry type = LOCAL
Protocol = TCPIP
Hostname = honcho
Service name = 523
Remote instance name =
System = HONCHO
Operating system type = WIN
```

```
DB2 LIST NODE DIRECTORY SHOW DETAIL
```

```
Node Directory

Number of entries in the directory = 1
```

```
Node 1 entry:
```

```
Node name = NODE1
Comment =
Directory entry type = LOCAL
Protocol = TCPIP
Hostname = honcho
Service name = 78787
Remote instance name = db2inst1
System = HONCHO
Operating system type = WIN
```

To change the system name from HONCHO to PLATO, you would issue the following commands to recatalog the above nodes with a new system name:

```
DB2 UNCATALOG NODE HONCHO
DB2 CATALOG ADMIN TCPIP NODE HONCHO REMOTE HONCHO SYSTEM PLATO OSTYPE WIN
DB2 UNCATALOG NODE NODE1
DB2 CATALOG TCPIP NODE NODE1 REMOTE HONCHO SERVER 78787
REMOTE_INSTANCE db2inst1 SYSTEM PLATO OSTYPE WIN
```

On restarting the Control Center, the system name is now displayed as PLATO. The system will still have a single instance (db2inst1) located under it in the object tree.

Related concepts:

- “About systems” in *Administration Guide: Planning*

Related tasks:

- “Cataloging database systems” on page 177


Related reference:

- “CATALOG TCPIP/TCPIP4/TCPIP6 NODE command” in *Command Reference*
- “LIST NODE DIRECTORY command” in *Command Reference*
- “UNCATALOG NODE command” in *Command Reference*

Getting help in the Control Center

Use the  toolbar icon or the **Help** menu to get help or additional information.

These are the types of help and information that you can get:

-  Opens the DB2 Information Center to enable you to search for help on tasks, commands, and other information on DB2 and IMS.

Help menu

Displays menu items for displaying the master index, general information about the Control Center, and keyboard help . This menu also provides links to:

- Tutorials available with DB2
- How to use the help
- The DB2 Information Center
- Product information.

Related concepts:

- “Features of the DB2 Information Center” in *Online DB2 Information Center*
- “Control Center overview” on page 376




Related tasks:

- “Keyboard shortcuts and accelerators (all centers)” in *Online DB2 Information Center*

Using advisors, wizards, and launchpads to perform tasks quickly and easily

The DB2 advisors, wizards, and launchpads are integrated into the DB2 administration tools. They assist you in completing administrative tasks by stepping you through the tasks.






You can open the following advisors, wizards, and launchpads from the Wizards window accessed from the Control Center **Tools**→**Wizards** menu:

-  Add Partitions launchpad
- Backup wizard
- Create Database wizard . See also the Create Your Own Database wizard, accessed from the Control Center or from First Steps.
- Create Table Space wizard
- Create Table wizard
- Design advisor
- Load wizard
-  Configuration advisor
- Restore wizard
- Configure Database Logging wizard
- Set Up Activity Monitor wizard
-  Set Up High Availability Disaster Recovery (HADR) Databases wizard

Depending on your selection, you will be prompted to select an instance, database, or table. Follow the instructions in the advisor, wizard, or launchpad to complete your task. Click the hyperlinks (underlined text), if any, to link to additional information.

The following wizards and launchpads are available from other parts of the DB2 product.

Also from the Control Center:

-  Add Partitions launchpad
-  Alter Database Partition Group wizard
- Configure Automatic Maintenance wizard
- Configure Multisite Update wizard
- Create Cache Table wizard
-  Drop Partition launchpad
-  Redistribute Data wizard
-  Storage Management Setup launchpad

For federated systems:

- Create Federated Objects wizard (this is also called the Create Nicknames wizard)
- Create XML Configuration File wizard (This wizard is part of in the WebSphere Federation Server SDK).

For OS/390 and z/OS systems:

- Create Cloning Session or Edit Cloning Session wizard
- Create Object Maintenance Policy wizard

For IMS:

- Query Database wizard
- Query Transaction wizard
- Update Database wizard
- Update Data Group wizard
- Update Transaction wizard

From the Health Center:

- Health Indicator Configuration launchpad
- Recommendation advisor
- Health Alert Notification Troubleshooting wizard

From the Replication Center:

- Replication Center launchpad
- Add Capture Control Server wizard
- Add Apply Control Server wizard
- Add Monitor Control Server wizard
- Add Q Capture Server wizard
- Add Q Apply Server wizard
- Add Server Information wizard

- Create Monitor wizard
- Create Q Capture Control Tables wizard
- Create Q Apply Control Tables wizard
- Create Q Subscriptions wizard
- Create XML publications wizard

Related concepts:

- “Control Center overview” on page 376

Wizard overviews

This section contains two examples of wizard overviews, accessed from the first page of the wizard.

Backing up data using the Backup wizard

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

Prerequisites:

To back up a database or a database partition, you must have SYSADM, SYSCTRL, or SYSMAINT authority.

Before you create a backup plan: If you change a database configuration file to enable rollforward recovery (using either LOGRETAIN or USEREXIT), you must take an offline backup of the database before it is usable.

Procedure:

To back up the objects in your database:

1. Open the Backup wizard: From the Control Center, expand the object tree until you find the database or table space object that you want to back up.

Note: You can select one or more table space objects to back up. Right-click on the object and select **Backup** from the pop-up

Note: The Backup wizard opens. To back up a database partition, open the Backup wizard from the database object. To back up a table space partition, open the Backup wizard from the table space object.

2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is available when you complete enough information for the wizard to back up the objects in your database.

Related concepts:

- “Backup overview” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “BACKUP DATABASE command” in *Command Reference*

Restoring data using the Restore wizard

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

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

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

Prerequisites:

- To restore a database or a database partition, you must have SYSADM, SYSCTRL, or SYSMAINT authority.
- To restore to a new database, you must have SYSADM or SYSCTRL authority.
- To restore a table space or table space partition, you must have SYSADM, SYSCTRL, or SYSMAINT authority.

Before you can restore a database you must have an exclusive connection; that is, no applications can be running against the database when the task is started. Once it starts, it prevents other applications from accessing the database until the restore is completed.

Procedure:

To restore the objects in your database:

1. Open the Restore wizard: From the Control Center, expand the object tree until you find the database or table space object that you want to restore. Right-click on the object and select **Restore** from the pop-up menu. The Restore wizard opens.

Note: To restore a database partition, open the Restore wizard from the database object. To restore a table space partition, open the Restore wizard from the table space object.

2. Complete each of the applicable wizard pages. Click the wizard overview link on the first page for more information. The **Finish** push button is available when you complete enough information for the wizard to restore the objects in your database.

Related concepts:

- “Restore overview” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “RESTORE DATABASE command” in *Command Reference*

Control Center object tree and details view

This section describes the Control Center’s object tree and details view, including how to add, display, and filter objects.

Expanding and collapsing the Control Center object tree

To expand and collapse the object tree, click the plus signs (+) and minus signs (-) next to objects in the object tree.

For example, look at the **All Systems** or the **All Databases** folder displayed on the object tree. If you click the plus sign (+) next to the **All Systems** folder, icons representing your local workstation and any remote systems connected to your local system are displayed. If you click the plus sign (+) next to a particular system icon, the **Instances** folder and any instances residing on that system are displayed. Similarly, if you click the plus sign (+) beside the **All Databases** folder, you will see the list of catalogued databases.

To collapse the object tree to the **All Systems** folder, click the minus sign (-) next to the **All Systems** folder. All of the objects under the **All Systems** folder are no longer displayed.

Related concepts:

- “Control Center overview” on page 376

Adding DB2 UDB for z/OS subsystems to the object tree

This topic refers collectively to the following products as DB2 UDB for z/OS:

- DB2 UDB for z/OS Version 8
- DB2 UDB for OS/390 and z/OS Version 7

To use the Control Center to manage DB2 UDB for z/OS subsystems, you must first add the subsystems to the object tree.

To open the Control Center:

- In Windows, click **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**.
- In Linux, open the **IBM DB2** folder on the desktop and click **Control Center**.

To add the subsystem to the object tree, configure a connection to the subsystem to enable you to access the objects in that subsystem. You will have to know the host system and subsystem names, the communication protocol, and the communication protocol parameters to use for connecting to the subsystem.

If you have installed the Configuration Assistant on the workstation where you are running the Control Center, you can use the Configuration Assistant to configure your workstation to access your DB2 subsystem. Otherwise, you must use the command line processor to add the subsystem.

Related concepts:

- “Control Center overview” on page 376

Adding DB2 federated system objects to the object tree

If you want to use the Control Center to manage your DB2 federated system DB2 federated system objects (such as wrappers, server definitions, nicknames, and cache tables) you must:

- Set the DB2 database parameter **FEDERATED** to **YES**.

- To access all data sources except the DB2 family of products and Informix[®], you must install WebSphere Federation Server on the server that will act as the federated server.
- Add the federated system objects to the object tree that are required for the data sources that you want to access.

To open the Control Center:

- In Windows, click **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**.
- In Linux, open the **IBM DB2** folder on the desktop and click **Control Center**.

To set the DB2 database parameter FEDERATED to YES, right-click the DB2 instance that you want to use as your federated database instance and select **Configure Parameters**. The DBM Configuration window appears. In the list of Environment parameters, change the FEDERATED parameter to YES and click **OK**.

To install WebSphere Federation Server, follow the steps that come with the WebSphere Federation Server software.

To add the federated system objects required for the data sources that you want to access, select the database that you want to use as your federated database and right-click the **Federated Database Objects** folder. Click **Create Federated Objects** to launch a wizard that guides you through the steps to create all of the necessary federated objects and adds the federated objects to the tree.

Related tasks:

- “Expanding and collapsing the Control Center object tree” on page 389

Adding DB2 systems and IMSplexes, instances, and databases to the object tree

Use the Control Center to add systems, instances, and databases to the object tree.

If you install a new computer system or create a new instance and you want to use the Control Center to perform tasks on it, you must add it to the object tree.

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

To add systems, instances, or databases to the object tree, you need SYSADM or SYSCTRL authority.

To open the Control Center:

- In Windows, click **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**.
- In Linux, open the **IBM DB2** folder on the desktop and click **Control Center**.

Expand the object tree to display the folder for the type of object (system, instance, or database) that you want to add. Right-click the folder. A pop-up menu of all of the available actions for the object opens. Click **Add**. The Add window opens.

For adding DB2 systems, instances, or databases in the Add window, you can access a list of existing remote systems, instances, or databases by clicking the **Discover** push button. This is not an option for adding IMS systems.

Related tasks:

- “Expanding and collapsing the Control Center object tree” on page 389

Refreshing objects in the objects tree and details view

To refresh the entire object tree, click **Refresh** in the View menu from the menu bar.

To update the view of the objects displayed in the contents pane, use one of the following methods:

- Click **View** -> **Refresh**.
- Right-click a folder or object and click **Refresh** in the pop-up menu.

The contents pane displays the contents of the object selected on the object tree.

Related concepts:

- “Control Center overview” on page 376

Deleting custom folders or objects in custom folders

To delete a custom folder from the Control Center’s object tree, right-click the custom folder that you want to delete and click **Delete** in the pop-up menu.

To delete an object within a custom folder, select it and click **Alter** in the pop-up menu. A system or database removed from a custom folder can still be found under the **All Systems** or **All Databases** folders, and any other custom folders where it has been placed.

Note: Either of the **All Systems** or **All Databases** folders might themselves be completely hidden by using the Control Center View window .

Related concepts:

- “Control Center overview” on page 376

Database unavailable status in the database details pane of the Control

You can use the Control Center’s details pane to view information about your databases. Selecting a database in the object tree or contents pane displays a summary of its state. In certain situations database information might be unavailable. Some reasons for this unavailability are described in the following table.

Table 19. Reasons for a database status of unavailable

Database status element	Possible reasons for unavailable status
Last backup	<ul style="list-style-type: none">• No backups have been performed for the database.• User does not have the required authority to access this information.
Size	<ul style="list-style-type: none">• Database is pre-Version 8.2.• User does not have the required authority to access this information.

Table 19. Reasons for a database status of unavailable (continued)

Database status element	Possible reasons for unavailable status
Capacity	<ul style="list-style-type: none"> • Database is pre-Version 8.2. • Database has multiple partitions. • User does not have the required authority to access this information.
Health	<ul style="list-style-type: none"> • Health monitor is not turned on. • Timing delay. There is approximately a 5 minute delay from the time a database is activated until its health status is available.
Maintenance	<ul style="list-style-type: none"> • Database is pre-Version 8.2.

Related concepts:

- “Control Center overview” on page 376

Displaying objects in the Control Center

Use the Control Center to display objects to enable you to work with them. Objects are displayed in the object tree and the contents pane .

To open the Control Center:

- In Windows, click **Start -> Programs -> IBM DB2 -> General Administration Tools -> Control Center**.
- In Linux, open the **IBM DB2** folder on the desktop and click **Control Center**.

To display objects in the object tree and contents pane, expand the object tree by clicking on the plus signs (+) next to objects. As you expand the object tree down from a particular object, the objects that reside in, or are contained in, that object are displayed underneath. At the lowest level of the tree, folders of objects (such as tables) that do not contain other objects are displayed.

Click an object (folder or icon) in the object tree. The objects that reside in, or are contained in, the selected object are displayed in the contents pane. Systems, subsystems, instances, and databases are displayed in both the object tree and the contents pane. Objects that do not contain other objects are displayed only in the contents pane. For example, when you click a **Tables** folder, all of the tables in the database are displayed in the contents pane.

Related concepts:

- “Control Center overview” on page 376

Displaying table information in the contents pane

Use the contents pane of the Control Center, to display table information. The view uses rollups to help you to quickly access the information and also allows greater flexibility in representing large amounts of complex data in table form. The view groups the key elements and allows you to name and save them for future use. It also allows you to group rows of a display together when they all share the same value in a specific column.

Use the **Overview by categories** rollup menu on the contents pane toolbar and the **View** menu to perform functions on the table data in the contents pane.



- The **Overview by categories** rollup menu allows you to switch between views.
- The **View** menu allows you to perform functions on the table data such as naming and saving the views; filtering, sorting, and customizing the columns; and printing and exporting.
- The filter, sort, and customize columns functions are also available by right-clicking any column heading.

Tasks:

- Naming or saving the contents of the details view
- Filtering the list of displayed columns or table data
- Sorting the list of displayed table data
- Customizing the list of displayed columns

Related concepts:

- “Control Center overview” on page 376

Obtaining Control Center diagnostic information

For Windows platforms, use the **db2cc -tf <file-name>** command to request Control Center diagnostic information as directed by DB2 Customer Service. The <file-name> file is created in the sqllib\tools directory. If an absolute path is specified, the trace is created in the specified path.

On AIX and Linux platforms, use the **db2cc -t** command. The output is displayed on the console. You might direct the output to a file.

For Windows platforms only, there is also a **db2cctrc** command for getting a Control Center trace. Use it as follows:

```
db2cctrc file1 [cc-options]
```

- “file1” is the file where Control Center trace output is written. If no path name is specified, this file is created in the tools directory under the sqllib directory.
- “[cc-options]” (optional) see the documentation for the **db2cc** command for a list of available options

Note that standard out and standard error information are written to the console.

To see diagnostic information, use the DB2 Trace facility .

Attention::

For performance reasons, only use these commands when directed to do so by DB2 Customer Service or by a technical support representative.

Related concepts:

- “Basic trace diagnostics” in *Troubleshooting Guide*
- “Diagnostic tools (Linux and UNIX)” in *Troubleshooting Guide*
- “Interpreting diagnostic log file entries” in *Troubleshooting Guide*


Related tasks:

- “Setting the diagnostic log file error capture level” in *Troubleshooting Guide*

Related reference:

- “db2trc - Trace command” in *Command Reference*
- “Diagnostic tools (Windows)” in *Troubleshooting Guide*

Finding objects in the contents pane

Use  in the contents pane toolbar, or click **Edit->Find** to find objects in the contents pane. Click a folder in the object tree to display the objects with which you want to work.

In the **Find string** field, type the character string that you want to find. The first object in the contents pane that meets the find search criteria is selected. If you want to find the next object meeting the find criteria, click **Edit->Find**.


Select the **Case sensitive** check box when searching for case sensitive strings.

Related concepts:

- “Control Center overview” on page 376

Filtering or pre-filtering objects

Use the Filter notebook to pre-filter, or the Filter window to filter, the list of displayed objects in the details view.

- The Filter notebook opens when you right-click any folder object in the object tree and click **Filter**. The pre-filter action alters the query used to retrieve objects from the database.
- The Filter window opens when you click **View->Filter** or when you click the  from the contents pane toolbar. The filter action filters the objects after they are retrieved from the database.

In the details view, the pre-filter and filter effects are cumulative. The same object is selected in the object tree so that its pre-filtered children appear in the filtered details view. That is, the number of filtered pre-filtered objects are less than or the same as those that are only pre-filtered.

To change the default for filtering when number of rows exceed the default value, see Setting startup and default options for the DB2 administration tools

Related concepts:

- “Control Center overview” on page 376

Related tasks:

- “Setting startup and default options for the DB2 administration tools” on page 436

Extending the Control Center

This section describes how you can extend the Control Center by adding new toolbar buttons including new actions, adding new object definitions, and adding new action definitions.

Introducing the plug-in architecture for the Control Center

You can extend the DB2 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 might 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 396
- “Guidelines for Control Center plugin developers” on page 395
- “Writing plugins as Control Center extensions” on page 397

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 sql11ib directory. Before V8, the db2plug.zip needed to be installed in the cc directory under the sql11ib 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 sql11ib 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 396
- "Writing plugins as Control Center extensions" on page 397

Related reference:

- "db2cc - Start control center command" in *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 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 might 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 might 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:

- “Guidelines for Control Center plugin developers” on page 395
- “Writing plugins as Control Center extensions” on page 397

Related reference:

- “db2cc - Start control center command” in *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 an 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:

- “Adding a menu item only to an object with a particular name” on page 402
- “Adding an example object under the folder” on page 405
- “Adding the alter action” on page 410
- “Adding the create action” on page 407
- “Adding the folder to hold multiple objects in the tree” on page 403
- “Adding the remove action with multiple selection support” on page 409
- “Creating a basic menu action” on page 399
- “Creating a basic menu action separator” on page 401
- “Creating a plugin that adds a toolbar button” on page 398
- “Creating sub menus” on page 401
- “Positioning the menu item” on page 400
- “Setting attributes for a plugin tree object” on page 406

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, a toolbar button is added, 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 396

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, new commands will be added 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 no other features other than the ability to add menu items to Control Center built-in objects are used (for example, the Database object in this example), most functions will return null or true. To specify that this object represents the DB2 database object, its type is specified as UDB_DATABASE, a constant in CCOBJECT. The class is named CCDatabase in this example, however class names should be as unique as possible, since there might 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 getObject method of your CCExtension should return an array containing an instance of CCDatabase as follows:

```
public CCOBJECT[] getObject () {
    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 a class that implements this interface is written.

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 DB2 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 396

Related tasks:

- "Adding a menu item only to an object with a particular name" on page 402
- "Creating a basic menu action separator" on page 401
- "Creating sub menus" on page 401
- "Positioning the menu item" on page 400

Positioning the menu item: **Procedure:**

When creating the basic menu item, the position of the menu item within the menu is not specified. 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 `CCEExtension`.

Change `CCDatabase` to add `Example2BAction` to the menu as follows:

```
public CCMenuAction[] getMenuActions () {
    return new CCMenuAction[] { new Example2AAction(),
                               new Example2BAction() };
}
```

Related tasks:

- “Adding a menu item only to an object with a particular name” on page 402
- “Creating a basic menu action” on page 399
- “Creating a basic menu action separator” on page 401
- “Creating sub menus” on page 401

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:

- “Adding a menu item only to an object with a particular name” on page 402
- “Creating a basic menu action” on page 399
- “Creating sub menus” on page 401
- “Positioning the menu item” on page 400

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:

- “Adding a menu item only to an object with a particular name” on page 402
- “Creating a basic menu action” on page 399
- “Creating a basic menu action separator” on page 401
- “Positioning the menu item” on page 400

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 it refers to a database, the system, instance and database names must be included in what is returned 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 399
- “Creating a basic menu action separator” on page 401
- “Creating sub menus” on page 401

- “Positioning the menu item” on page 400

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, the CCTreeObject is implemented instead of CCOBJECT so that plugin objects show up under Database in the Control Center tree. First 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 CColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public CMenuAction[] 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 might 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 396

Related tasks:

- “Adding an example object under the folder” on page 405
- “Adding the alter action” on page 410
- “Adding the create action” on page 407

- “Adding the remove action with multiple selection support” on page 409
- “Setting attributes for a plugin tree object” on page 406

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 might include new or changed child objects which might 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 Example3Folder 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, this was not done.

Related concepts:

- “Compiling and running the example plugins” on page 396

Related tasks:

- “Adding the alter action” on page 410
- “Adding the create action” on page 407
- “Adding the folder to hold multiple objects in the tree” on page 403
- “Adding the remove action with multiple selection support” on page 409
- “Setting attributes for a plugin tree object” on page 406

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 Example3Child(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 396

Related tasks:

- “Adding an example object under the folder” on page 405
- “Adding the alter action” on page 410
- “Adding the create action” on page 407
- “Adding the folder to hold multiple objects in the tree” on page 403

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 code shown above, a new class called CCOBJECTCollectionEvent is used 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, only the first object is cast in the Vector and addChild is called 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 () {
```

```

        return menuActions;
    }
    ...
}

```

Related concepts:

- “Compiling and running the example plugins” on page 396

Related tasks:

- “Adding an example object under the folder” on page 405
- “Adding the alter action” on page 410
- “Adding the folder to hold multiple objects in the tree” on page 403
- “Adding the remove action with multiple selection support” on page 409
- “Setting attributes for a plugin tree object” on page 406

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 might 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 396

Related tasks:

- “Adding an example object under the folder” on page 405
- “Adding the alter action” on page 410
- “Adding the create action” on page 407
- “Adding the folder to hold multiple objects in the tree” on page 403
- “Setting attributes for a plugin tree object” on page 406

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 396

Related tasks:

- “Adding an example object under the folder” on page 405
- “Adding the create action” on page 407
- “Adding the folder to hold multiple objects in the tree” on page 403
- “Adding the remove action with multiple selection support” on page 409
- “Setting attributes for a plugin tree object” on page 406

License Center

This section describes how to use the License Center, including how to add, change and remove licenses. It also describes how to view license information.

License Center overview

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

To open the License Center:

Click  in the Control Center. The License Center opens.

Tasks:

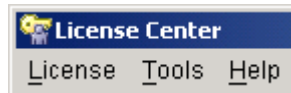
- Adding licenses
- Changing licenses and policies
- Viewing licensing information
- Viewing license policy information
- Viewing authorized user infraction information
- Viewing and resetting compliance details
- Removing licenses

Accessibility:

- Using the keyboard

The License Center interface:

The License Center interface has two elements that help you add and manage licenses.

Menu bar

The License Center menu bar contains the following menus:

License

Use this menu to add, change or remove licenses, to generate the compliance report, and to reset the compliance information. You can also use this menu to refresh all products, shut down all of the DB2 server administration tools, and exit from the License Center.

Tools

Use this menu to open any of the DB2 tools. Some of the functions in this menu are also available by clicking the icons on the toolbar. For more information, see Tools menu.

Help

Use this menu to display online help and product information, and to open the Information Center and the Tutorial. For more information, see Help menu.

Toolbar

Use the toolbar icons below the menu items to access other DB2 administration tools. For more information, see DB2 toolbar.

Related concepts:

- “Control Center overview” on page 376
- “License management” on page 64


Related tasks:

- “Adding licenses” on page 412
- “Changing licenses and policies” on page 413
- “Viewing license policy information” on page 414
- “Viewing licensing information” on page 413
- “Removing licenses” on page 416
- “Viewing and resetting compliance details” on page 415
- “Viewing authorized user infraction information” on page 415

Adding licenses

From the License Center, use the Add License window to add new licenses.

Procedure:

1. Open the Add License window: Click  in the Control Center to open the License Center. Select the system for which you want to add a new license. Select **License**→**Add**. The Add License dialog opens.
2. Select the license file (*.lic) that you want to add.

Related concepts:

- “License Center overview” on page 411

Changing licenses and policies


Use the Change License window to change the enforcement or license type policies.

Prerequisites:

To modify license policies in the License Center, you need SYSADM authority on the DB2 instance that contains the installed license.

Note: If you do not have SYSADM authority, the Select Instance window automatically displays, from which you can select an instance where you have SYSADM authority.

Procedure:

1. Open the License Center: Click  in the Control Center.
2. Select the system and the installed product for which you want to change enforcement or license type policies, as follows:
 - To change the enforcement policy, in the Enforcement policy pane, select “Hard stop” or “Soft stop”.
 - To change the license type policy, change the number of users for one or both of the following license types:
 - Concurrent users policy (available only for DB2 Connect Enterprise Server Edition): Controls and monitors the number of users that can connect simultaneously to a single DB2 server.
 - Authorized users policy: Controls and monitors users, by user ID, that are allowed to connect to a server. An authorized users policy is typically used in an environment with more than one DB2 server.
 - For DB2 Connect Enterprise Server Edition only, to change the number of concurrent users, activate the concurrent users policy and type the number of purchased licenses.


Related concepts:

- “License Center overview” on page 411

Viewing licensing information

Use the License Center to view details about any of your DB2 licenses.

Procedure:

1. Open the License Center: Click  in the Control Center.
2. Select a system name and an installed product for which you want to view licensing information.

License page

You must have a system selected for the License page to be enabled. The information displayed varies depending on the type of license and product you have installed.

Statistics page

This page is available only when a concurrent policy is enabled and the selected system and selected product have been used for DB2 activities.

Select the **Summary** radio button to view the statistical information in a text format. Or, select the **Graph** radio button to view the statistical information in a bar graph format.

Select the date on which the summary information begins and use the calendar control to select a start date. If you do not specify a date, all available data will be displayed.

Select the date on which the summary information ends and use the calendar control to select an end date.

You can refresh the statistical information currently displayed, retrieve statistical information, or view information regarding authorized user infractions.

Related concepts:

- “License Center overview” on page 411


Related tasks:

- “Viewing and resetting compliance details” on page 415
- “Viewing license policy information” on page 414
- “Viewing authorized user infraction information” on page 415

Viewing license policy information

Use the License Center to view details about your license policy. The license policy controls and monitors the number of users that can connect simultaneously to a single DB2 server.

Procedure:

1. Open the License Center: Click  in the Control Center.
2. Select the system name and the installed product for which you want to view details about your license policy.
3. On the License page, enabled only if a system is selected and if you have DB2 Connect Enterprise Server Edition installed, click the **Concurrent users policy** arrow to reveal the information in the other fields.

Related concepts:

- “License Center overview” on page 411

Related tasks:


- “Viewing licensing information” on page 413
- “Viewing and resetting compliance details” on page 415
- “Viewing authorized user infraction information” on page 415

Viewing authorized user infraction information

Use the License Center Details window to view information regarding authorized user infractions.

The Statistics page is available only when a User-based policy is enabled and the selected system and selected product have been used for DB2 activities. Statistics are generated during connects and disconnects after the database manager has been restarted.

Procedure:

1. Open the License Center: Click the  icon in the Control Center.
2. On the Statistics page, if enabled, view information about licensed and non-licensed users.

Related concepts:

- “License Center overview” on page 411


Related tasks:

- “Viewing and resetting compliance details” on page 415
- “Viewing license policy information” on page 414
- “Viewing licensing information” on page 413

Viewing and resetting compliance details

From the License Center, you can generate a compliance report that lists any features of the DB2 server that you may be using out of compliance with your current terms and conditions. Some functions are only available under license when purchased as part of a DB2 feature. You can remove the license violation warnings in the compliance report by purchasing the feature and installing the license file (.lic) shipped on the feature media.

Procedure:

1. Open the Generate Compliance Report window: Click  in the Control Center to open the License Center. Select **License→Generate Compliance Report**. The Generate Compliance Report window opens.
2. View the compliance details.
3. Optional: To reset the license usage information, select **License→Reset Compliance Report**.

Note: The reset option resets all license usage information for all products and instances installed within the selected install path. You cannot reset usage information selectively for a product or for a particular feature.

Related concepts:

- “License Center overview” on page 411

Related tasks:

- “Viewing authorized user infraction information” on page 415
- “Viewing license policy information” on page 414
- “Viewing licensing information” on page 413


Removing licenses

Use the License Center to remove a license.

Note: If you do not have SYSADM authority, the Select Instance window automatically displays, from which you can select an instance where you have SYSADM authority .

To remove DB2 licenses, you need SYSADM authority on the DB2 instance on which the license is installed.

To remove a license:

1. Open the License Center: Click  in the Control Center.
2. Select the system and the product from which the license is to be removed.
3. Select **License**→**Remove** and confirm your request.

Related concepts:

- “License Center overview” on page 411

Task Center and Journal

This section describes how to use the Task Center and Journal for scheduling, running, and viewing task information. It also describes how to manage contacts, saved schedules, success code sets, and task categories.

Task Center overview

Use the Task Center to schedule tasks, to run tasks, and to notify people about the status of completed tasks. Tasks are actions performed by the following types of scripts:

- DB2 command scripts, if the scripts contain DB2 commands
- OS command scripts, if the scripts contain operating system commands
- MVS shell scripts, if the scripts contain MVS commands to be run in a host environment, such as z/OS

You can also create grouping tasks to define actions based on the results of multiple tasks. Grouping tasks are unlike other tasks in the Task Center, because no command script is directly associated with a grouping task. Instead, a grouping task contains tasks that are already defined to the Task Center. The advantage of creating a grouping task is to create task actions that depend on the results of more than one task.

Task schedules are managed by a scheduler. The tasks are run on one or more systems, known as run systems. You define the conditions for a task to fail or succeed with a success code set. Based on the success or failure of a task, or group of tasks, you can run additional tasks, disable scheduled tasks, and perform related actions. You can also define notifications to send after a task completes. You can send an e-mail notification to people in your contacts list, or you can send a notification to the Journal.

From the Task Center, you can also open other centers and tools to help you with other administrative tasks.

To open the Task Center, click  on the Control Center toolbar.

Prerequisites:

To use the Task Center, you must select a scheduler system that will work with the Task Center. The Task Center uses the system clock of the scheduler to determine when to start tasks. To select a scheduler system, from the Task Center, select a system in the **Scheduler System** field.

When you log on to the Task Center, you are logging on to the scheduler that you select. You must log on every time you start the Task Center.

To grant or revoke privileges for a task, you must be the creator of the task. To create, alter, or delete a task, you must have write authority for the task. To run a task, you must have run authority for the task.

Tasks:

You can perform the following tasks from the Task Center:

- Create or edit tasks
- Run tasks immediately
- Manage contacts
- Manage task categories
- Manage saved schedules
- Manage success code sets
- Change the default notification message

The Task Center interface:


The Task Center interface consists of three elements that help you to customize your view of the list of tasks and to navigate the Task Center efficiently.

Menu bar



Use the menu bar to work with objects in the Task Center, open other administration centers and tools, and access online help.

Contents pane

Click  to open the Task Center. Use the contents pane to display and work with system and database objects. The contents pane displays the tasks that are in the current view.

Contents pane toolbar




Use the toolbar below the contents pane to tailor the view of tasks in the contents pane to suit your needs. You can also select these toolbar functions in the Edit menu and the View menu.

Accessing custom controls with the keyboard

You can use the keyboard to access controls found on the graphical user

interface (GUI) interface. The following two controls are unique to the DB2 family of products. To access these controls using the keyboard:

- For the ellipsis , press **Tab** until the button is selected; then press **Enter**.
- For the Date field, press **Tab** until the field is selected; then type the date in the field.

Related concepts:

- “Control Center overview” on page 376
- “Journal overview” on page 418

Related tasks:

- “Changing the default notification message” on page 423
- “Creating or editing a task” on page 425
- “Managing contacts” on page 428
- “Managing saved schedules” on page 429
- “Managing success code sets” on page 430
- “Managing task categories” on page 431
- “Running tasks immediately” on page 421

Journal overview

Use the Journal notebook to view historical information about tasks, database actions and operations, messages, and notifications. The Journal is the focal point for viewing all available historical information generated within the Control Center, as compared to the **Show Result** option from the Task Center, which shows only the latest execution results of a task.

To sort the records shown in each of the notebook pages, click the column headings.

To open the Journal, click  on the Control Center toolbar.

Prerequisites:

To access the Journal, you must have access to the DB2 tools catalog database.

Task History page:


Use this page to view the task history records for each of the available scheduler systems and to analyze the execution results. For example:

- You might want to examine the status of weekly backup tasks.
- You might want to get the execution results of a specific execution of a task, such as a task that runs periodically to capture a snapshot of a database system. The results for each execution of this task can be viewed in the Journal.

From the **Refresh options** field, select the amount of time between automatic page refreshes. The default option is **No automatic refresh**.

To delete records, highlight the records that you want to delete, right-click and select **Delete** from the pop-up menu.

Database History page:

Use this page to view historical records of database recovery for each of the databases in the drop-down list. Click  to select a system, instance, and database. For partitioned environments, you must also select a database partition.

Messages page:

Use this page to view the message records issued by the DB2 administration tools on the local system. To delete the message records, highlight the records that you want to delete, right-click and select **Delete** from the pop-up menu. Alternatively, you can use the Selected menu to remove only the selected records or all records.

Notification Log page:

Use this page to view the notification log records for the selected instance. You can customize the filtering options and criteria. The default is to display the last 50 records of all notification types. If you select either **Read from specified record to end of the file** or **Read records from specified range**, and if the settings are set to overwrite old records, the log record numbers are not reused. Therefore, selecting Start record 1 and End record 100 does not guarantee seeing anything in the notification log if the log has been looping. Note that the columns and column headings change depending on your selection.

Related concepts:


- “Task Center overview” on page 416
- “Health Center overview” in *System Monitor Guide and Reference*
- “Control Center overview” on page 376

Related tasks:

- “Creating a database for the DB2 tools catalog” on page 424
- “Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration” on page 96

Enabling scheduling settings in the Task Center

Use the Advanced Scheduling Settings window to enable scheduling settings in the Task Center. Scheduling information is stored in the DB2 tools Catalog database .

Click  on the Control Center toolbar to open the Task Center to view the current settings.

Procedure:

To enable schedule settings in the Task Center:

1. From the **Enabling Scheduling Function** group box of your current window or notebook, select the system where you want to create the DB2 tools catalog database, and click **Create New**.
2. On the window that opens, follow the instructions in Scheduling a task.

You can use the Scheduler Settings page of the Tools Settings notebook to set the default scheduling scheme. Note that if you set a default scheduling scheme, you can still override it at the task level.

Related concepts:

- “Scheduler” on page 420
- “Task Center overview” on page 416

Related tasks:

- “Creating a database for the DB2 tools catalog” on page 424
- “Scheduling a task” on page 422
- “Setting the default scheduling scheme” on page 449
- “Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration” on page 96

Scheduler

The scheduler is a DB2 system that manages tasks. This component of the DB2 Administration Server (DAS) includes the tools catalog database, which contains information that the Task Center uses. When you schedule a task, the Task Center uses the system clock of the scheduler to track when the tasks on the scheduler need to run.

The Task Center displays the list of cataloged systems or databases that have active schedulers. You must select a scheduler system to work with the Task Center. When you log on to the Task Center, you are logging on to the scheduler system that you select. You must log on every time you start the Task Center.

Related concepts:

- “Task Center overview” on page 416

Related tasks:

- “Scheduling a task” on page 422
- “Enabling scheduling settings in the Task Center” on page 419
- “Setting the default scheduling scheme” on page 449

Success code sets

The Task Center uses success code sets to evaluate the success or failure of a task. Success code sets let you specify the return codes or range of return codes that you will accept to consider the task a success. Return codes outside the range that you specify indicate a failed task.

The Task Center evaluates the success of every statement of a DB2 script. If any statement fails, the entire task fails.

If you do not specify a success code set, a return code of 0 is considered a success; all others are failures. The following rules apply when you specify a success code set:

- The success code set can only have one greater than (>) condition, where the associated code must be greater than or equal to (>=) any less than (<) condition that is specified. For example, if you specify (> 5) or (< 0), the error codes 0, 1, 2, 3, 4, and 5 mean the task failed. You cannot specify (> 5) or (< 6), as this includes all numbers.
- The success code set can only have one less than (<) condition, where the associated code must be less than or equal to (<=) any greater than (>) condition

that is specified. For example, if you specify (< 0) or (> 5), the error codes 0, 1, 2, 3, 4, and 5 mean the task failed. You cannot specify (< 5) or (4 >), because this includes all numbers.

- There can be zero or more unique equality (=) conditions. Tasks with return codes that match an equality are considered a success.

The following example shows how to create a success code set:

Assume that you want to run a DB2 script. Also assume that this DB2 script consists of more than one SQL or XQuery statement (that is, the script contains multiple lines), and you know that each statement in the script returns an SQLCODE. Because some SQL or XQuery statements can return non-zero SQLCODES that do not represent error states, you must determine the set of non-error SQLCODES that any of the SQL statements in the script can return. For example, assume that the following return codes all indicate successful execution of the SQL or XQuery statements in the script. That is, if any of the following conditions are met, execution of the script continues:

```
RC > 0 OR RC = -1230 OR RC = -2000
```

You would define the success code set as shown in Table 20:

Table 20. Example of a success code set

Condition	SQLCODE
>	0
=	0
=	-1230
=	-2000

Related tasks:

- “Managing success code sets” on page 430


Running tasks immediately

In the Task Center, you can run one or more tasks immediately or schedule them to run at a later time. Even if you run a scheduled task immediately, it continues to run on existing schedules that might be associated with it, if they are enabled.

Prerequisites:

To run a task, you must have run authority for the task.

Procedure:

1. Open the Run Now window: Click  on the Control Center toolbar to open the Task Center. In the Task Center, select one or more tasks and click **Selected->Run Now**. The Run Now window opens.
2. Click **Use notifications** to use the notifications for the task.
3. Click **Use task actions** to use the task actions for the task.
4. If you are running an OS script task:
 - a. Include the parameters specified on the Run properties page of the Task Properties notebook.
 - b. Specify parameters to send.

- c. Include the name of the run system.
 - d. Specify additional parameters. If you checked **Override default parameters**, the parameters to send are shown in the **Parameter list preview** field.
5. Type a user ID with authority to run the task, and the password associated with the user ID.

Related concepts:

- “Task Center overview” on page 416
- “Tasks and required authorizations” on page 608

Related tasks:

- “Enabling scheduling settings in the Task Center” on page 419
- “Selecting users and groups for new tasks” on page 427

Scheduling a task

Whenever you create a task, you have the option of running it immediately or scheduling it to run later. For the latter, the script is saved in the Task Center, and all execution information is automatically saved in the Journal.

Procedure:

Use the Schedule page of various wizards and notebooks to indicate whether you want to run a selected task immediately, or schedule it to run later:

- To run the task immediately, without creating a task in the Task Center or saving the task history to the Journal, select **Run now without saving task history**.
- To create a task for generating the DDL script and saving it in the Task Center, select **Create this as a task in the Task Center**. Then, specify the task information and options:

- Specify the name of the system on which you want to run the task. This system must be online at the time the task is scheduled to run.
- Select the system where you want to store the task and the schedule information, in the **Scheduler system** drop-down box.

This system will store the task and notify the run system when it is time to run the task. The drop-down list contains any system that is cataloged and has the scheduler enabled. The scheduler system must be online so that it can notify the run system.

If the DB2 tools catalog is on a remote system, you will be asked for a user ID and password in order to connect to the database.

- Optional: If you want to select a different scheduling scheme, click **Advanced**. The Advanced Schedule Settings window opens where you can select between server scheduling or centralized scheduling.
- To save the task in the Task Center, but not actually run the task, select **Save task only**.
- To save the task in the Task Center and run the task now, select **Save and run task now**.
- To save the task to the Task Center, and schedule a date and time to run the task later, specify **Schedule task execution**. The **Change** button is enabled. Click **Change**. A window opens where you can enter the date and time at which to run the task.

The **Details** group box displays the schedule information you selected.

- To run a task in the Task Center you must supply a user ID and password. Type the user ID and password that you want to use to run the task.

You can use the Scheduler Settings page of the Tools Settings notebook to set the default scheduling scheme. Note that if you set a default scheduling scheme, you can still override it at the task level.

Related concepts:

- “Scheduler” on page 420
- “Task Center overview” on page 416

Related tasks:

- “Setting the default scheduling scheme” on page 449
- “Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration” on page 96

Changing the default notification message

Use the Edit Message window from the Task Center to change the default notification message that is sent to selected contacts to notify them of actions that they need to take.

Procedure:


1. Open Edit Message window. To open the Edit Message window, click  in the Control Center. The Task Center opens. In the Task Center, click **Task->Set Default Notification Text**. The Edit Message window opens.
2. Type the name or ID to identify the sender of the message. DB2 appends the @ hostname to the specified name or ID. The sender is the name that the e-mail message reports as the person who sent the message.
3. Type the subject line and text of the e-mail message. You can use the tokens in Table 21, which the Task Center recognizes and replaces with actual values in the e-mail message.

Table 21. Tokens for subject lines and e-mail messages

Token	Description
&Categories	The categories associated with the task.
&Completionstatus	The completion status of the task. This value depends on the success code set associated with the task.
&Description	The description of the task.
&Duration	The length of time that the run system took to complete the task from start to finish.
&End	The date and time when the task completed.
&Howinvoked	The method used to invoke the task.
&Name	The name of the task.
&Owner	The name of the owner of the task.
&Returncode	The final return code of the task.
&Runpartitions	The partitions on which the task ran.
&Runsystem	The name of the system on which the task ran.
&Schedulersystem	The name of the system on which the task is scheduled.
&Start	The date and time when the task began running.

Table 21. Tokens for subject lines and e-mail messages (continued)

Token	Description
&Type	The task type: DB2 script, OS script, MVS shell script, or grouping task.
&Userid	The user ID for the task.

4. Select the system parameters to include in the e-mail message.
5. Select to send the same e-mail message to pagers that is sent to all contacts, or select to send a different e-mail message to pagers. For the second option, the contact must be identified as having a pager e-mail address. For this option:
 - a. Type the subject line and text of the e-mail message that is sent to pagers. You can use the tokens in Table 21 on page 423, which the Task Center recognizes and replaces with actual values in the e-mail message.
 - b. Select the system parameters to include in the e-mail message that is sent to pagers.

Related concepts:

- “Task Center overview” on page 416

Creating a database for the DB2 tools catalog

Use the Create New Tools Catalog window to create the DB2 tools catalog on a *cataloged* system that currently has no metadata storage. The tools catalog is a set of tables in a database that is used by the Task Center to store task definitions, schedules, and task history information. You can choose to use an existing database or create a new one. Creating this database also enables Task Center, which provides support for a more complex scheduling scheme for managing different tasks and scripts.

Procedure:

- Open the Create New Tools Catalog window: Click **Create New** from the **Enabling Scheduling Function** group box of your current window, wizard, or notebook. For instance, click **Create New** from the Schedule page of the Tools Settings notebook.
- Select the instance where the tools catalog is to be created, and type the tools catalog schema name.
- Specify whether to create a new database or use an existing one.
- Select whether to force all applications for instance restart and whether to activate the tools catalog when it is created.

Troubleshooting tips:

If you receive a -567 return code when trying to create a new tools catalog, try running the **db2admin setid** command and stop and start the DAS. This error indicates that an invalid user ID or password has been submitted.

For more information on how to stop and start the DAS, see:

- dasdrop - Remove a DB2 Administration Server Command
- dasauto - Autostart DB2 Administration Server Command

Related concepts:

- “Tools catalog” in *Administration Guide: Planning*

Related reference:

- “dasdrop - Remove a DB2 administration server command” in *Command Reference*
- “dasauto - Autostart DB2 administration server command” in *Command Reference*
- “CREATE TOOLS CATALOG command” in *Command Reference*
- “DROP TOOLS CATALOG command” in *Command Reference*

Creating or editing a task

From the Task Center you can create a tasks for each DB2 script, OS script, or MVS shell script that you want to run. You can also create or edit grouping tasks that contain more than one task. Tasks that you include in grouping tasks continue to run on enabled schedules.

Grouping tasks are unlike other tasks in the Task Center, because no command script is directly associated with a grouping task. Instead, a grouping task contains tasks that are already defined to the Task Center. The advantage of creating a grouping task is to create task actions that depend on the results of more than one task. For example, you can place three backup tasks in a grouping task, then run a reorganization task only if all three backup tasks are successful. If any of the tasks in the grouping task fails, the grouping task is considered a failure.


Prerequisites:

Before creating a task, ensure that you have specified a scheduler system. The Task Center uses the system clock of the scheduler to determine when to start tasks.

When you log on to the Task Center, you are logging on to the scheduler that you select. You must log on every time you start the Task Center.

To create or edit a task, you must have write authority for the task.

Procedure:**To create or edit a task::**


1. Open the New Task notebook: Click  on the Control Center toolbar to open the Task Center. In the Task Center, select **Task**→**New**, or right-click anywhere in the task details view, and click **New**. The New Task notebook opens.
2. Select the type of task to create:
 - DB2 command script if the script contains DB2 commands
 - OS command script if the script contains operating system commands
 - MVS shell script if the script contains MVS commands to be run in a host environment, such as z/OS
 - Grouping task to place multiple tasks into the grouping task.
3. Optional: Select a task category. Categorizing tasks helps keep your list of tasks organized.
4. Select the system on which the task will run.
5. Specify the DB2 instance where the script will run. If the task will run on multiple DB2 partitions, select the partitions on which the task will run.
6. Refer to the appropriate path in this step for the type of task that you are creating, based on your selection in the **Type** field:

- Specify run properties for a DB2 command script:
 - a. On the Run properties page, select or create a success code set. Indicate if the task should stop immediately after receiving a failing return code. Indicate if any of the generated return codes is a failure. If you do not specify this, only the final return code is considered.
 - b. On the Command Script page, type the DB2 script or import an existing script from a file. Indicate the termination character; DB2 scripts that contain multiple statements must use this character to separate the statements. Type the name of the directory from which the script will run. Specify the full path name. Do not specify a mapped network drive.
 - Specify run properties for an OS command script:
 - a. On the Run properties page, specify a script interpreter and indicate the parameters that you want to pass to the script interpreter. Optional: Select or create a success code set. Success code sets specify the conditions required for the task to be successful. If you do not specify a success code set, only a return code of 0 is considered successful.
 - b. On the Command page, type the script or import an existing script and select the name of the directory on the run system where the script will run. Specify the full path name. Do not specify a mapped network drive.
 - Specify run properties for an MVS shell script:
 - a. On the Run properties page, select or create a success code set. Success code sets specify the conditions required for the task to be successful. If you do not specify a success code set, only a return code of 0 is considered successful.
 - b. On the Command page, type the script.
 - Select tasks for a grouping task:
 - a. On the Group page, determine which tasks to include in the grouping task.
 - b. Use the arrows to move tasks from the list of available tasks to the list of selected tasks. The selected tasks are members of the group.
7. Optional: Create a schedule. You can create a task without creating a schedule or specifying an existing schedule. However, to run the task, you must schedule the task as follows:
 - a. Specify a date and time for tasks on this schedule to begin running, including a repeating schedule, if applicable. Add the schedule to the list of schedules. You can continue adding to this list.
 - b. Optional: Save the schedule for reuse with other tasks. When you use a saved schedule with multiple tasks, you can update the schedule in the Saved Schedules window. To start the process in which the task runs, you must type the user ID and password, and enable the task before the first start time.
 - c. Optional: Enable the task. The task only runs when it is enabled. You must enable the task before the first start time. For non-recurring schedules, if you enable the task after its first start time, the task will not run. For recurring schedules, you can enable the task *after* the specified (first) start time as long as it is not past the end date, if specified.
 8. Optional: Create notifications. You can create a task without creating notifications. However, a notification is needed to inform people about the status of completed tasks. On the Notification page, specify the condition upon which to send the notification and the kind of notification to send when


the condition is met. You can add multiple notifications. For example, specify one notification to be sent if the task succeeds and another if the task fails. The notification is sent only when the notifications are enabled.

9. **Optional: Create task actions.** A task action determines which action to take after a task completes. You can specify different actions depending on the success or failure of the task. If you do not create a task action, no action is taken after the task completes. On the Task Actions page, specify the condition upon which to run the task action and select the kind of action to take when the condition is met. You can add multiple task actions. For example, you can run multiple tasks if the task succeeds or run a backup task if the task fails. This example requires only two task actions: one that names the tasks to run if the preceding task is successful, and one that names the tasks to run if the preceding task fails. The task actions are run only when the task actions are enabled. Specify the amount of time to wait between each retry of the task.
10. **Authorize the users who need to access this task with the appropriate access level.**

To select tasks to include in a grouping task::

1. **Open the New Task notebook:** Click  on the Control Center toolbar to open the Task Center. In the Task Center, select **Task->New**, or right-click anywhere in the task details view, and click **New**. The New Task notebook opens.
2. **On the Group page, select the tasks to include in the grouping task.** Use the arrow buttons to move them to the selected tasks. The selected tasks are members of the group.

To view statistics and the status of completed tasks::

1. **Open the Task Center:** Click  on the Control Center toolbar. The Task Center opens.
2. **Click Task->Show Progress.** The Show Progress window opens, in which you can view statistics and the status of completed tasks.

Related concepts:

- “Scheduler” on page 420

Related tasks:

- “Running tasks immediately” on page 421

Selecting users and groups for new tasks

Use the Select Users and Groups window to select available users or groups that will have authority to execute the new task being created.

To select users and groups for new tasks::

1. **Open the Select Users and Groups window.** To open the Select Users and Groups window:
 - a. **Open the Task Center and select Task->New** or right-click anywhere in the task details view and click **New**. The New Task notebook opens.
 - b. **Click the Security tab.**
 - c. **Click Change.** The Select Users and Groups window opens.
The **Available users and groups** table shows all available users and groups defined to the Task Center.

The **Selected users and groups** table shows any users and groups already selected.

2. Select one or more users or groups from the **Available users and groups** list, and click > to move them to the **Selected users and groups** list.
3. Click **OK** to save the specified information and return to the Security page of the New Task notebook.

Related concepts:

- “Control Center overview” on page 376

Related tasks:

- “Granting database authorities to new groups” on page 529
- “Granting database authorities to new users” on page 529

Managing contacts

Contacts are records of names and e-mail addresses that are stored in the Task Center. You use and manage the list of contacts like an address book. You can also create groups of contacts, which makes it easier to manage notification lists because you only need to update the group definition once to change the notification list for all tasks. When a notice is sent to the group, each member of the group receives the notice. Contact groups can include other contact groups. Other DB2 tools, such as the Health Center, can also use this contacts list. You can select one or more contacts from the list to receive e-mail notifications about a task after it completes.


The following tasks are part of managing contacts:

- Adding a contact
- Adding a contact group
- Changing a contact or contact group
- Viewing related contacts and contact groups
- Removing a contact or contact group

Restrictions:

To send notifications to contacts in the Task Center, DAS must be configured to send e-mail messages.

Procedure:

1. To specify a valid SMTP server for sending e-mail messages:
 - a. On the windows specified in following steps, click the **SMTP server** push button, and specify a valid SMTP server on the window that opens.
2. To add a contact:
 - a. Open the Add Contact window: Click  on the toolbar.
 - b. Click **Add Contact**.
 - c. Type the name of the contact. The name identifies the person who receives the e-mail notification message.
 - d. Type the full e-mail address of the contact (for example, userid@domain.com).
 - e. Optional: Specify if the e-mail message will be displayed on a pager that receives e-mail. You must click this check-box to use pager messages.

- f. Optional: Type a description of this contact.
3. To add a contact group:
 - a. Open the Contact Group window: From the Tools menu, click **Contacts**. The Contact Group window opens.
 - b. Click **Add Group**.
 - c. Type the name of the contact group.
 - d. Optional: Type a description for the contact group.
 - e. Move the available contacts to the **Selected Contacts** list.
4. To change a contact or contact group:
 - a. Open the Change Contact or Change Contact Group window: In the Task Center, click **Task->Contacts**. Select a contact or contact group to change. Click **Change**. Depending on your selection, either the Change Contact or Change Contact Group window opens.
 - b. Make your changes.
5. To view contact groups that contain a contact, or to remove a contact or contact group:
 - a. Open the Contacts window: In the Task Center, click **Task->Contacts**.
 - b. To see a list of the groups in which the contact is a member: Select a contact and click **Show Related**.
 - c. To remove a contact or contact group: Select it, right-click and select **Remove** from the pop-up menu. When prompted, confirm your selection.

Related concepts:

- “Task Center overview” on page 416

Managing saved schedules


In the Task Center, *schedules* can be saved and used to run one or more tasks. These are called *saved schedules*. The following tasks are part of managing saved schedules:

- Adding saved schedules
- Changing saved schedules
- Removing saved schedules
- Viewing saved schedules

Prerequisites:

To create, alter, or delete a saved schedule, you must have write authority for the saved schedule.

Procedure:

1. Click  on the Control Center toolbar to open the Task Center.
2. To see the current list of saved scheduled, select **Task->Saved Schedules**. The Saved Schedules window opens.
3. To add new schedules that can be reused for other tasks, click **Add**. The Add Saved Schedules notebook opens. (You can also open this notebook from the Schedule page of the New Task notebook by clicking the **Save List of Schedules**.) When you use a saved schedule with multiple tasks, you make updates to the schedule in one place, that is, from the Add Saved Schedules notebook, as follows:

- On the Saved Schedule page, type a name for the schedule and optionally a description.
 - On the Schedule page, specify a date and time for this schedule, including a repeating schedule, if applicable. Add the schedule to the list of schedules. You can continue adding to the list.
 - On the Security page, authorize the users who need to access this schedule with the appropriate access level.
4. To make a change to a saved scheduled, select it and click **Change**. The Change Saved Schedules notebook opens where you can make your changes.
 5. To delete any of the saved schedules, select them and click **Remove**. Then Confirm your request.
 6. To view the list of tasks for the selected saved schedule, click **Show Related**. The Show Related notebook opens, showing all related tasks and schedules.

Related tasks:

- “Creating or editing a task” on page 425

Managing success code sets


The Task Center uses success code sets to evaluate the success or failure of a task. Success code sets let you specify the return codes or range of return codes that you will accept to consider the task a success. Return codes outside the range that you specify indicate a failed task. For more information, see Success code sets.


The following tasks are part of managing success code sets:

- Adding success code sets
- Changing success code sets
- Selecting success code sets
- Viewing tasks that use the same success code set
- Removing success code sets

All Task Center users can create, alter, or delete success code sets.

Procedure:

1. Click  on the Control Center toolbar to open the Task Center.
2. To add a success code set:
 - a. Open the Add Success Code Set window: From the Task Center, click **Task->Success Code Sets**. From the Success Code Sets window, click **Add**. The Add Success Code Set window opens.
 - b. Type the name of the success code set and, optionally, a description.
 - c. Specify an operator: =, >, or <.
 - d. Type an integer representing a return code to go with the condition operator. The condition and code together make up the success code. If any of the expressions are met when the task completes, the task is successful.
 - e. Click **Add** to add the success code to the list of success codes.
 - f. Optional: Define additional success codes to add to the set.
3. To change a success code set:
 - a. Open the Change Success Code Set window: In the Task Center, click **Task->Success Code Sets**. The Success Code Set window opens. Select a success code set to change. Click **Change**. The Change Success Code Set window opens.

- b. Make your changes. You can change the name and description of the success code set. You can also add, change or remove success codes from the List of Success Codes.
4. To select a success code set, to view tasks that use the same success code set, or to remove a success code set:
 - a. Open the Select Success Code Set window: In the Task Center window, select a task. Click **Selected→Edit Task Properties**. Click the Run properties tab. In the **Success Code Set** field, click . The Select Success Code Set window opens.
 - b. To select a success code set: Select it from the list of available success code sets.
 - c. To view tasks that use the same success code set: Click **Show Related**.
 - d. To remove one or more success code sets: Select them, right-click and select **Remove** from the pop-up menu. When prompted, confirm your selection.

Related concepts:

- “Success code sets” on page 420
- “Task Center overview” on page 416


Managing task categories


Categorizing tasks lets you group them into relationships to help you manage them more efficiently. A task can belong to multiple categories. For example a payroll task that is run at the end of each month might be in the Monthly category and the Payroll category. You specify the category names and tasks that belong to the category in the Add Task Category window.

The following tasks are part of managing task categories:

- Adding a new category
- Changing a task category
- Selecting a task category
- Viewing tasks in the same category
- Removing a category

Procedure:

1. Open the Task Categories window: Click  on the Control Center toolbar to open the Task Center. In the Task Center, click **Task→Task Categories**. The Task Categories window opens.
2. To add a task category:
 - a. Open the Add Task Category window: In the Task Categories window, click **Add**. The Add Task Category window opens.
 - b. Type the name of the category, and optionally, a description of this category.
3. To change a task category:
 - a. In the Task Categories window, select a task category to change. Click **Change**. The Change Task Category window opens.
 - b. Make your changes to the name or description of the task category.
4. To select a task category:

- a. Open the Select Task Categories window: In the Task Center window, select a task and click **Selected→Edit Task Properties**. The Edit Task Properties notebook opens. On the Task page, click . The Select Task Categories window opens.
 - b. In the **Available task categories** list, select one or more categories to associate with the task.
 - c. Click the arrow to move the selected tasks to the **Selected Task Categories** list.
 - d. If the category name does not exist: Type the name in the **New task category** field. Click the arrow to move the task to the **Selected Task Categories** list.
5. To view tasks that are in the same category: In the Task Categories window, select a category, and click **Show Related**.
 6. To remove one or more categories: In the Task Categories window, select the categories to be removed, right-click and select **Remove** from the pop-up menu. When prompted, confirm your selection.

Related concepts:

- “Task Center overview” on page 416

Tools Settings

This section describes how to use the Tools Settings notebook to set up various properties and options. It also describes how to set up the startup and default options for the DB2 administration tools.

Tools Settings overview

Use the Tools Settings notebook to customize settings and set properties for DB2 administration tools, including documentation settings, font, and color. Some of the pages display only after you install the centers for which they apply.

General page:

Use this page specify whether the local DB2 instance should be automatically started when the DB2 tools are started, whether to use a statement termination character, and whether to set filtering when the maximum number of rows is exceeded from a display sample contents request.

For more information, see [Setting the server administration tools startup property](#).

Documentation page:

Use this page to specify whether hover help and infopop help features in the DB2 administration tools should display automatically, and also to specify the location from which the contextual help is accessed at the instance level.

For more information, see [Setting up access to DB2 contextual help and documentation](#).

Fonts page:

Use this page to change the font in which text and menus appear in the DB2 administration tools. From the fields available, select the font size and color in which you want the menus in the DB2 administration tools to appear.

Note: Some changes will not take effect until the Control Center is restarted. If you have chosen a font color that will not show up on the background color on your system, DB2 will temporarily override the font color that you have chosen and select a font color that will show up. This system override will not be saved as part of your user profile.

For more information, see [Changing the fonts for menus and text](#).

OS/390 and z/OS page:

Use this page to set column headings and define the online and batch utility execution options for OS/390 and z/OS objects. Defaults are provided for some of the options. For more information, see "Estimating column sizes" in [Setting DB2 UDB OS/390 and z/OS utility execution options](#).

For the **Optimize grouping of objects for parallel utility execution** option, see Example 1 for online and Example 2 for batch. If this option is not selected, objects are grouped according to the order in which they were selected, with the maximum number of objects in each group. See Example 1 and Example 2.

For the **Specify the Maximum number of objects to process in parallel for online execution** option, see Example 1. For the **Maximum number of jobs to run in parallel for batch execution** and **Maximum number of objects per batch job** options, see Example 2.

For more information, see [Setting DB2 UDB OS/390 and z/OS utility execution options](#).

Health Center Status Beacon page:

Use this page to specify the type of notification you will receive when an alert is generated in the Health Monitor. You can be notified through a pop-up message or with the graphical beacon that displays on the lower-right portion of the status line for each DB2 center, or using both methods of notification.

For more information, see [Enabling or disabling notification using the Health Center Status Beacon](#).

Scheduler Settings page:

Use this page to set the default scheduling scheme. Select **Server Scheduling** if you want task scheduling to be handled by the scheduler that is local to the database server, if the scheduler is enabled on that system. Select **Centralized Scheduling** if you want the storage and scheduling of tasks to be handled by a centralized system, in which case you need to select the centralized system from the **Centralized Scheduler** list. To enable another scheduler, select a system and click **Create New** to open a window in which you can create a database for the DB2 Tools Catalog on a cataloged system. If the system you want is not cataloged, you must catalog it first.

For more information, see [Setting the default scheduling scheme](#).

Command Editor page:

Use this page to specify how you will generate, edit, execute, and manipulate SQL and XQuery statements, IMS commands, and DB2 commands and work with the resulting output. These settings affect commands, SQL statements and XQuery statements on DB2 databases, z/OS and OS/390 systems and subsystems, and IMSplexes.

For more information, see [Setting Command Editor options](#).

IMS page:

Use this page to set your preferences when working with IMS. You can set preferences for using wizards, syntax support, results, and the length of your command history.

For more information, see [Setting IMS options](#).

Related concepts:

- “Features of the DB2 Information Center” in *Online DB2 Information Center*

Setting the server administration tools startup property

Use the General page of the Tools Settings notebook to specify that the local DB2 instance should be automatically started when the DB2 tools are started. You can also use this page to specify whether to automatically expand the **All Databases** folder at start up.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the General page, select the **Automatically start local DB2 on tools startup** check box.

Related tasks:

- “Setting startup and default options for the DB2 administration tools” on page 436
- “Starting and stopping the DB2 administration server (DAS)” on page 94
- “Starting the server DB2 administration tools” on page 369

Setting a command statement termination character

Use the General page of the Tools Settings notebook to specify that a character will be used to terminate statements in command scripts. Command scripts are used in the Command Editor and Task Center.

Note: If you specify a statement termination character, you cannot use the backslash (\) character to continue statements in command scripts.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the General page, select the **Use statement termination character** check box. Optional: Type a character that will be used as the statement termination character in the entry field. The default character is a semicolon (;).

Command scripts are used in the Command Editor and Task Center. If you specify a statement termination character, you cannot use the backslash (\) character to continue statements in command scripts.

Related concepts:

- “Command Editor overview” in *Online DB2 Information Center*

Related tasks:

- “Executing commands and SQL statements using the Command Editor” in *Online DB2 Information Center*
- “Setting Command Editor options” on page 449

Setting up access to DB2 contextual help and documentation

Use the Documentation page of the Tools Settings notebook to specify:

- Whether to automatically display hover help and infopop help features in the DB2 administration tools
- Where you want to access the DB2 Information Center from so that you can view DB2 contextual help and documentation.

To open the Tools Settings notebook, click  on the DB2 toolbar. Click the **Documentation** tab.

To indicate whether hover help will be automatically displayed, select or clear the **Automatically display hover help** check box. The default setting is for the hover help to be automatically displayed.

To indicate whether infopops will be automatically displayed, select or clear the **Automatically display infopops** check box. The default setting is for the infopops to be automatically displayed. If you clear this check box so that infopops are not automatically displayed, you can still press **F1** to see the infopop for a particular field or control.

In the **Documentation location** fields, specify for this instance where to access the DB2 Information Center from:

- To access the DB2 Information Center on the IBM Web site, use the default values.
- To access the DB2 Information Center installed on an intranet server, or on your own computer, specify the host name and port number of the server or computer.

Important:

The documentation location values that you specify on this page update the DB2 DB2_DOCHOST and DB2_DOCPORT profile registry variables that control how requests for DB2 documentation are handled for this instance only. If you want to change the settings for *all* instances on this computer, or if you want to change them for a single user session, follow the instructions in Setting the location for accessing the DB2 Information Center.

To have the **Documentation location** values take effect, including resetting the default values, click **Set** and restart the center in which you are working.

Related concepts:

- “Features of the DB2 Information Center” in *Online DB2 Information Center*
- “DB2 Information Center installation options” in *Quick Beginnings for DB2 Servers*
- “Environment variables and the profile registry” on page 65

Related tasks:

- “Setting the location for accessing the DB2 Information Center” in *Troubleshooting Guide*
- “Declaring, showing, changing, resetting, and deleting registry and environment variables” on page 68
- “Troubleshooting problems with the DB2 Information Center running on local computer” in *Troubleshooting Guide*

Related reference:

- “Miscellaneous variables” in *Performance Guide*

Setting startup and default options for the DB2 administration tools

Use the General page of the Tools Settings notebook to specify DB2 administration tools start up properties and customize your Control Center.

To open the Tools Settings notebook, click  on the DB2 toolbar.

The following properties and defaults can be set:

Automatically start local DB2 on tools startup

Select this check box automatically start the local DB2 instance when the DB2 tools are started.

Automatically expand the All Databases folder

Select this check box to automatically expand the **All Databases** folder whenever the Control Center is launched.

Use statement termination character

Select this check box to change the default statement termination character, and specify a new value in the field provided.

Set filtering when number of rows exceeds [value]

Select this check box to change the default number of rows displayed before filtering is required. The default is 500 rows. If more than the specified number of rows is returned, a Filter window automatically opens, allowing you to filter the data.

Limit the number of table rows fetched for editing [value]

Select this check box to change the default number of rows that you can edit at any one time, in the field provided. The default is 100 rows. If there are less rows than the specified value, the **Fetch More Rows** push button is disabled

Click **Customize the Control Center** push button to switch between the basic, advanced, or custom views .

Related tasks:

- “Finding service level information about the DB2 administration tools environment” on page 370

- “Setting the server administration tools startup property” on page 434
- “Shutting down server DB2 administration tools” on page 369
- “Starting the server DB2 administration tools” on page 369

Changing the fonts for menus and text

Use the Fonts page of the Tools Settings notebook to change the font in which text and menus appear in the DB2 administration tools.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the Fonts page, select the font size and color in which you want the menus and text in the DB2 administration tools to appear.

Troubleshooting tips:

- Some changes will not take effect until the Control Center is restarted.
- If you have chosen a font color that will not show up on the background color on your system, DB2 will temporarily override the font color that you have chosen and select a font color that will show up. This system override will not be saved as part of your user profile.

Related reference:


- “DB2 Help menu” on page 375
- “DB2 Tools menu” on page 374

Setting DB2 UDB OS/390 and z/OS utility execution options

Use the OS/390 and z/OS page of the Tools Settings notebook to set column headings and define utility execution options.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the OS/390 and z/OS page, select the **Use system catalog column names as column headings** check box to match the column headings in the contents pane of the Control Center, if applicable, to the column names defined in the system catalogs of DB2 UDB for OS/390 or z/OS. For derived columns, that is, columns of which the values are not selected directly from the system catalog, this option will not have any effect. If you do not select this option, all the column headings in the contents pane will be displayed in translated form in the current language selection.

Optional: Select the **Edit options each time utility runs** check box to have the opportunity to modify the utility execution options each time a utility is executed. The **Online execution utility ID template** field shows the current template for identifying DB2 for OS/390 and z/OS utilities. You can type an identifier, keep the displayed value, or open a Change Online Execution Utility ID Template window to select from a list of symbolic variables by clicking . The template can consist of literals and symbolic names. Symbolic names start with an ampersand (&) and end in a period (.).

The resolved length of the utility identifier can be no longer than 16 characters. If you do not create your own identifier, the Control Center generates a default utility ID from the date and timestamp. The default format is *CCMMddhhmmss*,

where: *CC* is the Control Center, *MM* is the month (01-12), *dd* is the day (01-31), *hh* is the hour, *mm* is the minute, and *ss* is the second. When parallel execution is being used, this name is truncated to 9 characters, and a 7-character unique identifier is added by the system.

Optional: For online, select **Continue online execution or batch job if error is encountered** if you want any of the parallel threads started by the Control Center to start execution of a utility against an unprocessed object. This would occur if executing a utility in any concurrently running thread, or in the same thread, resulted in an error (a DSNUTILS return code of 8). If not selected, no more calls will be made to DSNUTILS once an error is found in any thread.

For batch, select **Continue online execution or batch job if error is encountered** if you want the next step of a job generated by the Build JCL or Create JCL function to be executed if the step immediately before has returned an error executing an utility (a return code of 8). Unlike with online execution, there exists no dependency between jobs (the next job with the same jobname would also start regardless of an error in the previous job with the same jobname). If not selected, the job generated by the Build JCL or Create JCL function will terminate when one of the steps executing a utility has returned an error (a return code of 8 or higher).

Optional: For online, select **Optimize grouping of objects for parallel utility execution** if you want the set of objects to be grouped into a number of parallel threads that are constrained by the setting **Maximum number of objects to process in parallel for online execution**. With this setting, you can minimize the overall execution time, use fewer parallel threads to achieve the shortest overall processing time, and optimize usage of system resource. See Example 1 below.

For batch, select **Optimize grouping of objects for parallel utility execution** if you want the set of objects to be grouped into a number of parallel threads (jobs) that are constrained by the setting **Maximum number of jobs to run in parallel for batch execution**. With this setting, you can minimize the overall execution time, use the fewest concurrent jobs to achieve the shortest overall processing time, and optimize usage of system resource. The maximum number of steps (executions of the utility) per job is limited by the setting **Maximum number of objects per batch job**. See Example 2 below.

If this option is not selected, objects are grouped according to the order in which they were selected, with the maximum number of objects in each group. See Example 1 and Example 2 below.

Specify the **Maximum number of objects to process in parallel for online execution**. The default is 10, the maximum value allowed is 99. This number is used as the maximum when using the optimizer to group objects, and applies only to online execution. If 1 is specified, then objects are not processed in parallel, but are processed sequentially. If optimization has not been selected, then this value specifies exactly how many threads there will be. See Example 1 below.

Specify the **Maximum number of jobs to run in parallel for batch execution**. The default is 10, the maximum value allowed is 99. This number is used as the maximum when using the optimizer to group objects, and applies only to batch, not to online execution. If 1 is specified, then objects are not processed in parallel, but instead are processed sequentially. If optimization has not been selected, then this value specifies exactly how many concurrent batch jobs there will be. See Example 2 below.

Specify the **Maximum number of objects per batch job**. The default is 10, the maximum value allowed is 255. This number is used as the maximum when using the optimizer to group objects, and applies only to batch. If optimization has not been selected, then this value specifies how many steps (one step per object) there will be in each batch job. See Example 2 below.

Example 1: Online execution: How objects are assigned to threads:

RUNSTATS is requested to be run against a set of index objects (IX1, IX2, IX3 PART 1, IX3 PART2) where IX3 is a partitioned index and the **Maximum number of objects to process in parallel for online execution** is set to 4.

The optimizer estimates that RUNSTATS on IX1 takes 10 times longer than on all other objects.

When optimization is selected

If optimization is enabled, the optimizer would only come up with 2 threads:

Thread 1 would run RUNSTATS against IX1.

Thread 2 would run RUNSTATS against IX2, IX3 PART 1 and IX3 PART 2 sequentially.

More threads would not result in a shorter overall execution, since Thread 1 will take longer than Thread 2.

When optimization is not selected

If optimization is disabled, 4 threads would be used in this example.

Example 2: Batch execution: How objects are assigned to jobs and job steps:

RUNSTATS is requested to be run against a set of index objects (IX1, IX2, IX3 PART 1, IX3 PART2, IX4, IX5, IX6, IX7, IX8, IX9, IX10) where IX3 is a partitioned index and the **Maximum number of jobs to run in parallel for batch execution** is set to 2 and the **Maximum number of objects per batch job** is set to 3.

The optimizer estimates that RUNSTATS on IX1 takes 10 times longer than on all other objects.

When optimization is selected

The following JCL would be created for the user:

```
//JOB1 JOB....
//STEP1 EXEC...
//..... RUNSTATS INDEX IX1
//JOB2 JOB.... //STEP1 EXEC...
//..... RUNSTATS INDEX IX2
//STEP2 EXEC...
//..... RUNSTATS INDEX IX3 PART 1
//STEP3 EXEC...
//..... RUNSTATS INDEX IX3 PART 2
//JOB2 JOB.... //STEP1 EXEC...
//..... RUNSTATS INDEX IX4
//STEP2 EXEC...
//..... RUNSTATS INDEX IX5
//STEP3 EXEC...
//..... RUNSTATS INDEX IX6
```

```

//JOB2 JOB....
//STEP1 EXEC...
//..... RUNSTATS INDEX IX7
//STEP2 EXEC...
//..... RUNSTATS INDEX IX8
//STEP3 EXEC...
//..... RUNSTATS INDEX IX9
//JOB2 JOB....
//STEP1 EXEC...
//..... RUNSTATS INDEX IX10

```

Since the utility execution on IX1 is expected to take as long as RUNSTATS on all the remaining 10 indexes and partitions together, RUNSTATS on IX1 is run in a separate job, while all the other RUNSTATS are run sequentially in JOB2. Note that a higher parallelism would not reduce the overall execution time of the workload. The optimizer chooses the least required parallelism to complete the workload in the shortest time.

When optimization is not selected

The following JCL would be created for the user:

```

//JOB1 JOB....
//STEP1 EXEC...
//..... RUNSTATS INDEX IX1
//STEP2 EXEC...
//..... RUNSTATS INDEX IX3 PART 1
//STEP3 EXEC...
//..... RUNSTATS INDEX IX4
//JOB2 JOB....
//STEP1 EXEC...
//..... RUNSTATS INDEX IX2
//STEP2 EXEC...
//..... RUNSTATS INDEX IX3 PART 2
//STEP3 EXEC...
//..... RUNSTATS INDEX IX5
//JOB1 JOB....
//STEP1 EXEC...
//..... RUNSTATS INDEX IX6
//STEP2 EXEC...
//..... RUNSTATS INDEX IX8
//STEP3 EXEC...
//..... RUNSTATS INDEX IX10
//JOB2 JOB....
//STEP1 EXEC...
//..... RUNSTATS INDEX IX7
//STEP2 EXEC...
//..... RUNSTATS INDEX IX9

```

Only the last set of jobs will be created with less than the maximum number of objects allowed. The list of objects will be assigned sequentially, alternating by jobs and steps, as shown in the example above.

Related tasks:

- “Adding DB2 UDB for z/OS subsystems to the object tree” on page 389

DB2 for z/OS health monitor

This section describes how to use DB2 for z/OS health monitor, including how to view recommended actions, health alert summaries, and health alert objects.

DB2 UDB for z/OS health monitor overview

On the z/OS system, the DB2 UDB for z/OS health monitor is started as a task for each DB2 subsystem to be monitored or on a dedicated member of a data sharing group.

The DB2 UDB for z/OS health monitor triggers the evaluation of object maintenance policies at scheduled times and intervals, as defined in the policy. The object maintenance policies are created using the DB2 Control Center's Create Object Maintenance Policy wizard. During each policy evaluation, the criteria for recommending maintenance is checked against the thresholds set in the object maintenance policy to determine the need for object maintenance, that is, whether COPY, REORG, RUNSTATS, STOSPACE, ALTER TABLESPACE, or ALTER INDEX are required, and to identify restricted states, such as CHKP, on table space, index, and storage group objects where applicable. When objects are identified to be in alert state during policy evaluation, the policy health alert contacts are notified at their e-mail addresses or pager numbers. The list of health alert contacts for each DB2 subsystem is defined in and managed from the Control Center.

A snapshot of the evaluation schedule for the policies, which is used by the health monitor to determine when to trigger policy evaluations, is initially taken by the health monitor when it is started. This schedule snapshot is refreshed at the refresh time specified when the health monitor was started, or when the health monitor receives a refresh command. Any change to the evaluation schedule of a policy is picked up by the health monitor when the schedule refresh occurs.

The health monitor is started and stopped from the console, using the MVS system **START** and **STOP** commands, respectively.

A sample cataloged procedure (DSNHMONP) that starts a DB2 health monitor, and a sample cataloged procedure (DSNHMONA) that starts multiple DB2 health monitors within an MVS system or Parallel Sysplex, are placed in a procedure library by the installation job DSNTIJHM.

Views, tables, data sets, cataloged procedures, stored procedures, user-defined functions, and the result set table, which are used by the db2 health monitor or the related tasks listed below, are created and installed by the installation jobs DSNTIJCC and DSNTIJHM. DSNTIJCC and DSNTIJHM are shipped with FMIDs JDB771D and JDB881D.

Policy Evaluation Log:

Policy evaluations triggered by the DB2 health monitor are logged in the table DSNACC.HM_EVAL_LOG. An entry is logged when a policy evaluation starts and when a policy evaluation ends. Log entries are kept for 7 days, after which they will be deleted from the table. The DB2 view DSNACC.HM_ALERT_PO_EV, which was created on this table by the DSNTIJCC installation job, can be used to display all policies whose last evaluation iteration was not successful.

Related tasks:

- “Starting, stopping and refreshing the DB2 UDB for z/OS health monitor” on page 442
- “Viewing health alert summaries” on page 446
- “Viewing health alert objects” on page 447
- “Viewing, submitting, and saving recommended actions” on page 443

Starting, stopping and refreshing the DB2 UDB for z/OS health monitor

On the z/OS system, the DB2 UDB for z/OS health monitor is started as a task for each DB2 subsystem to be monitored or on a dedicated member of a data sharing group.

- To start a DB2 health monitor, issue the following **START** MVS system command:

```
S membername, DB2SSN=ssid, JOBNAME=HMONssid, TRACE=trace, REFRESH=nn
```

TRACE and REFRESH parameters are optional.

membername

Specifies a procedure library member that is executed to start the DB2 health monitor, that is, DSNHMONP. This cataloged procedure is created by the DSNTIJHM installation job.

ssid

Specifies the name or identifier of the DB2 subsystem to be monitored.

trace

Specifies the trace flag. Possible values are:

- ON - Turn on trace. Trace records are written to SYSOUT
- OFF - Do not turn on trace

The default is OFF.

nn

Specifies the hour (using a 24-hour clock) when the health monitor refreshes the evaluation schedule snapshot it uses to trigger policy evaluations. The default is 22.

- To start multiple DB2 health monitors, issue the following **START** MVS system command:

```
S membername
```

membername

A procedure library member that is executed to start multiple DB2 health monitors, that is, DSNHMONA.

Note: Before starting multiple DB2 health monitors with one **START** command using DSNHMONA, the HMONPARM data set specified in the DSNHMONA proc must be populated with the list of subsystems to be monitored. The cataloged procedure and the data set are created by the DSNTIJHM installation job.

- To refresh the policy evaluation schedule snapshot used by the DB2 health monitor to determine when to trigger policy evaluations, issue the following **MODIFY** MVS system command:

```
F HMONssid, APPL=REFRESH
```

ssid

Name or identifier of the DB2 subsystem that the DB2 health monitor you're refreshing is monitoring.

- To stop a DB2 health monitor, issue the following **STOP MVS** system command:

```
STOP HMONssid or P HMONssid
ssid
```

Name or identifier of the DB2 subsystem that the DB2 health monitor you're stopping is monitoring.

Related concepts:

- "DB2 UDB for z/OS health monitor overview" on page 441

Related tasks:

- "Viewing health alert summaries" on page 446
- "Viewing health alert objects" on page 447
- "Viewing, submitting, and saving recommended actions" on page 443

Viewing, submitting, and saving recommended actions

To view, submit, and save the actions recommended for alert objects identified during policy evaluation, call the DB2 stored procedure `SYSPROC.DSNACCHR`, which is created by the `DSNTIJCC` installation job. `DSNACCHR` is a stored procedure which determines the recommended actions for alert objects identified during policy evaluation and generates a JCL job that will execute the recommended actions.

The following syntax diagram shows the SQL CALL statement for invoking `DSNACCHR`. Because the linkage convention for `DSNACCHR` is `GENERAL WITH NULLS`, if you pass parameters in host variables, you need to include a null indicator with every host variable. Null indicators for input host variables must be initialized before you execute the CALL statement.

Syntax:

```
▶▶CALL DSNACCHR (—query-type,—health-ind,—policy-id,—work-set,——————▶
▶[dataset-name,] [member-name,] [save-opt,] trace-flag,—————▶
  [NULL] [NULL] [NULL]
▶—job-id,—jobname,—jcl-proc-time,—trace-flag,—last-statement,—————▶
▶—return-code,—error-msg )—————▶▶
```

query-type

Specifies what you want to do with the actions recommended for objects identified to be in alert state during policy evaluation. Possible values are:

- 0 - View recommended actions on alert objects as a JCL job
- 1 - Submit the JCL job that executes the recommended actions on alert objects
- 2 - Submit the JCL job that executes the recommended actions on alert objects, and put the job on the hold queue
- 3 Save recommended actions on alert objects as a JCL job in a library member

query-type is an input parameter of type `INTEGER`.

health-ind

Specifies the type of alert that DSNACCHR includes in the JCL job. Possible values are:

- RS - Restricted State
- EX - Extents Exceeded
- RR - REORG Required
- CR - COPY Required
- RT - RUNSTATS Required
- SS - STOSPACE Required

health-ind is an input parameter of type VARCHAR(4).

policy-id

Specifies an object maintenance policy. *policy-id* is an input parameter of type VARCHAR(7).

work-set

Specifies the work set of an object maintenance policy that identified the alert objects that DSNACCHR includes in the JCL job. This work set must be identified with the policy and type of alert specified in the parameters *policy-id* and *health-ind*. *work-set* is an input parameter of type INTEGER.

dataset-name

Specifies a fully qualified partitioned data set (PDS) or partitioned data set extended (PDSE) name. This value must be specified if *query-type* is 3. *dataset-name* is an input parameter of type VARCHAR(44).

member-name

Specifies a member of the partitioned data set (PDS) or partitioned data set extended (PDSE) specified in the *dataset-name* parameter where the object maintenance JCL job will be saved. This value must be specified if *query-type* is 3. *member-name* is an input parameter of type VARCHAR(8).

save-opt

Specifies how to save the object maintenance JCL job. This value must be specified if *query-type* is 3. Possible values are:

- R - Replace
- A - Append
- NM - New member

save-opt is an input parameter of type VARCHAR(2).

trace-flag

Specifies whether tracing will be turned on or off. Possible values are:

- Y - Turn trace on
- N - Turn trace off

trace-flag is an input parameter of type CHAR(1).

job-ID

When *query-type* is 1 or 2, specifies the job ID of the submitted job. *job-id* is an output parameter of type VARCHAR(8).

jobname

When *query-type* is 1 or 2, specifies the name of the submitted job. *jobname* is an output parameter of type VARCHAR(8).

jcl-proc-time

Specifies the time request was processed. *jcl-proc-time* is an output parameter of type TIMESTAMP.

last-statement

When DSNACCHR returns a severe error (return code 12), this field contains the SQL statement that was executing when the error occurred. *last-statement* is an output parameter of type VARCHAR(2500).

return-code

The return code from DSNACCHR execution. Possible values are:

- 0 - DSNACCHR executed successfully
- 12 - DSNACCHR terminated with severe errors. The *error-msg* parameter contains a message that describes the error. The *last-statement* parameter contains the SQL statement that was executing when the error occurred.

return-code is an output parameter of type INTEGER.

error-msg

When DSNACCHR returns a severe error (return code 12), this field contains error messages, including the formatted SQLCA. *error-msg* is an output parameter of type VARCHAR(1331).

DSNACCHR returns one result set when the *query-type* parameter is 0. The result set contains the JCL job generated by DSNACCHR. The DSNACCHR result set table is created by the DSNTIJCC installation job. Table 22 shows the format of the result set.

Table 22. DSNACCHR result set format

Column name	Data type	Description
JCLSEQNO	INTEGER	Sequence number of the table row (1,...,n)
JCLSTMT	VARCHAR(80)	Specifies a JCL statement

Related concepts:

- “DB2 UDB for z/OS health monitor overview” on page 441

Related tasks:

- “Viewing health alert objects” on page 447
- “Starting, stopping and refreshing the DB2 UDB for z/OS health monitor” on page 442
- “Viewing health alert summaries” on page 446

Viewing health alert summaries

The HEALTH_OVERVIEW function returns information from the Health Alert Summary VSAM KSDS data set as a DB2 table. This data set is created by the DSNTIJHM installation job. The Health Alert Summary data set contains information about the state of the DB2 health monitor and alert summary statistics for every DB2 subsystem previously or currently monitored by the health monitor on that MVS system or Parallel Sysplex. These information are returned to the client with a row for each DB2 subsystem and alert recommendation.

The result of the function is a DB2 table with the following columns:

ip-addr

The IP address of the DB2 server. This is a column of type VARCHAR(40).

db2-ssid

The subsystem identifier of the DB2 subsystem. This is a column of type VARCHAR(4).

health-ind

The type of alert. Possible values are:

- RS - Restricted State
- EX - Extents Exceeded
- RR - REORG Required
- CR - COPY Required
- RT - RUNSTATS Required
- SS - STOSPACE Required
- PO - Failed Policy Evaluation
- HM - Health Monitor State

health-ind is a column of type VARCHAR(4).

host-name

The fully qualified domain name of the DB2 server. This is a column of type VARCHAR(255).

summary-stats

The state of the DB2 health monitor if *health-ind* is 'HM'. Possible values are:

- 0 Health monitor is not started
- 1 Health monitor is started
- -1 Health monitor state is unknown

Otherwise, the total number of alert objects with the alert type specified in *health-ind*. This is a column of type INTEGER.

alert-state

The state of the alert specified in *health-ind*. Possible values are:

- 5 - Alarm
- 4 - Attention
- 3 - Warning
- 0 - Normal

alert-state is always 0 when *health-ind* is 'HM'. This is a column of type INTEGER.

The external program name for the function is HEALTH_OVERVIEW, and the specific name is DSNACC.DSNACCHO. This function is created by the DSNTIJCC installation job.

Example: Find the total number of alert objects requiring COPY for the DB2 subsystem 'ABCD':

```
SELECT SUMMARYSTATS FROM TABLE (DSNACC.HEALTH_OVERVIEW()) AS T
WHERE DB2SSID = 'ABCD'
AND HEALTHIND = 'CR';
```

Related concepts:

- “DB2 UDB for z/OS health monitor overview” on page 441

Related tasks:

- “Viewing health alert objects” on page 447
- “Starting, stopping and refreshing the DB2 UDB for z/OS health monitor” on page 442
- “Viewing, submitting, and saving recommended actions” on page 443

Viewing health alert objects

Alert objects identified during the last successful iteration of a policy evaluation are saved in these alert object repository tables, depending on their object type:

- DSNACC.HM_MAINT_TS for table spaces
- DSNACC.HM_MAINT_IX for indexes
- DSNACC.HM_MAINT_SG for storage groups

DB2 creates a number of views on these alert object repository tables. The views and alert object repository tables are created by the DSNTIJCC installation job. Table 23 lists the tables on which each view is defined and the view descriptions. All view names and table names have the qualifier DSNACC.

Table 23. Views on health alert objects

View Name	On Table	View Description
HM_ALERT_TS_RS	HM_MAINT_TS	Displays all table spaces in restricted state
HM_ALERT_TS_EX	HM_MAINT_TS	Displays all table spaces whose extents have exceeded a user-specified limit
HM_ALERT_TS_RR	HM_MAINT_TS	Displays all table spaces that require REORG
HM_ALERT_TS_CR	HM_MAINT_TS	Displays all table spaces that require COPY
HM_ALERT_TS_RT	HM_MAINT_TS	Displays all table spaces that require RUNSTATS

Table 23. Views on health alert objects (continued)

View Name	On Table	View Description
HM_ALERT_IX_RS	HM_MAINT_IX	Displays all indexes that are in restricted state
HM_ALERT_IX_EX	HM_MAINT_IX	Displays all indexes whose extents have exceeded a user-specified limit
HM_ALERT_IX_RR	HM_MAINT_IX	Displays all indexes spaces that require REORG
HM_ALERT_IX_CR	HM_MAINT_IX	Displays all indexes that require COPY
HM_ALERT_IX_RT	HM_MAINT_IX	Displays all indexes that require RUNSTATS
HM_ALERT_SG_SS	HM_MAINT_SG	Displays all storage groups that require STOSPACE

Related concepts:

- “DB2 UDB for z/OS health monitor overview” on page 441

Related tasks:

- “Starting, stopping and refreshing the DB2 UDB for z/OS health monitor” on page 442
- “Viewing health alert summaries” on page 446
- “Viewing, submitting, and saving recommended actions” on page 443

Enabling or disabling notification using the Health Center Status Beacon

Use the Health Center Status Beacon page of the Tools Settings notebook to specify the type of notification you will receive when an alert is generated in the Health Monitor. You can specify whether you want to be notified through a pop-up message or with a graphical beacon that displays on the lower-right portion of the status line for each DB2 Center, or using both methods of notification.

You can also specify that you do not want to receive notification using the Health Center status beacon.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the Health Center Status Beacon page, the check boxes on the Health Center Status Beacon page are enabled by default. Do the following:

- To enable notification through a pop-up message only, select the **Notify through pop-up message** check box and deselect the **Notify through status line** check box. When you select this method, a DB2 message window indicates that there are outstanding alerts.
- To enable notification using a status line graphical health beacon only, select the **Notify through status line** check box and deselect the **Notify through pop-up message** check box. When you select this method, a text message indicating that there are outstanding alerts and an graphical health beacon display on the Status line in each center.
- To disable notification, deselect the **Notify through pop-up message** and **Notify through status line** check boxes.

Related concepts:

- “About health indicators” in *Administration Guide: Planning*
- “Health alerts” in *Administration Guide: Planning*

Related reference:

- “Reading the contents of the health indicator settings fields” in *Online DB2 Information Center*

Setting the default scheduling scheme

Use the Scheduler Settings page of the Tools Settings notebook to set the default scheduling scheme. Note that if you set a default scheduling scheme, you can still override it at the task level.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the Scheduler Settings page, select **Server Scheduling** if you want scheduling to be handled by the scheduler that is local to the database server, if the scheduler is enabled on that system. Select **Centralized Scheduling** if you want the storage and scheduling of tasks to be handled by a centralized system, in which case you need to select the centralized system from the Centralized Scheduler list.

If you select **Centralized Scheduling**, select the centralized system from the **Centralized Scheduler** drop-down list. To enable another scheduler, select a system and click **Create New** to open a window in which you can create a database for the DB2 Tools Catalog on a cataloged system. If the system you want to use is not cataloged, you must catalog it first.

Related concepts:

- “Scheduler” on page 420
- “Task Center overview” on page 416

Related tasks:

- “Enabling scheduling settings in the Task Center” on page 419
- “Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration” on page 96

Setting Command Editor options

Use the Command Editor page of the Tool Settings notebook to set your preferences when working with the Command Editor. There are options that affect command and SQL statement execution and the resulting output.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the Command Editor page, set the **Execution and history** fields:

- Select **Automatically commit SQL statements** to have any changes made by SQL statement execution to take effect immediately.
- Select **Stop execution if errors occur** to have processing stop when there are errors.
- Select **Limit the number of elements stored in command history** to control the amount of command and statement execution history that appears in the

Command History window. Specify the size. If you do not select this option, all command and statement history will appear.

- Select **Log command history to file** to save command and statement execution history to a file and specify the file and location.

Set the **Output** options:

- Select **Limit the number of lines displayed in output** to control the amount of information that appears in the output section of the Commands page of the Command Editor notebook. Specify the number of lines. If you do not select this option, all processing lines will appear.
- Select **Enable wrapping of output text** to control the display of the information that appears in the output section of the Commands page of the Command Editor notebook. If you do not select this option, line wrapping will not occur and significant scrolling might be required to view the information.
- Select **Log output to file** to save output information to a file and specify the file name and location.
- Select **Do not display SQLCODE or SQLSTATE** to suppress the appearance of SQLCODEs and SQLSTATEs for executed statements in the output. Select **Display SQLCODE** to have the SQLCODE for executed statements appear in the output. Select **Display SQLSTATE** to have the SQLSTATE for executed statements appear in the output.

Related concepts:

- “Command Editor overview” in *Online DB2 Information Center*

Related tasks:

- “Executing commands and SQL statements using the Command Editor” in *Online DB2 Information Center*

Setting IMS options

Use the IMS page of the Tool Settings notebook to set your preferences when working with the IMS. You can set options that affect how you work with IMS commands and view the command results.

To open the Tools Settings notebook, click  on the DB2 toolbar.

On the IMS page, check **Enable IMS syntax support** to assist you when entering type-2 IMS commands in the Command Editor. When syntax support is enabled, lists of keywords are automatically displayed as you enter a command.

Check **Launch available wizards by default** to have the command wizard open initially from the Control Center. If you uncheck this option, command windows are opened.

Check **Automatically display IMS results** to have your command results automatically displayed. Check **Display results in a separate window** to have your command results always displayed in a separate window.

Select how many commands you want to keep in your IMS command results history.

Related concepts:

- “Control Center overview” on page 376

Related tasks:

- “Adding DB2 systems and IMSplexes, instances, and databases to the object tree” on page 390

Visual Explain

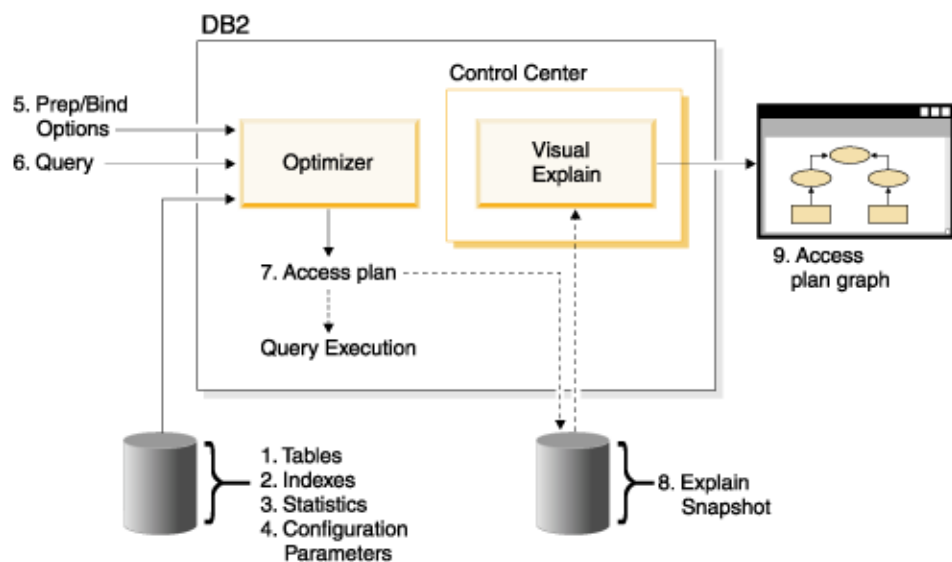
This section describes how to use Visual Explain to tune your SQL and XQuery statements.

Visual Explain overview

Visual Explain lets you view the access plan for explained SQL or XQuery statements as a graph. You can use the information in the graph to tune your queries by performing the following tasks:

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

The following illustration shows the interaction between the DB2 optimizer and Visual Explain invoked from the Control Center. (Broken lines indicate actions that are required for Visual Explain.)



To learn how to use Visual Explain, you can work through the scenarios in the Visual Explain Tutorial.

Prerequisites::

- To dynamically explain SQL or XQuery statements, you will need a minimum of INSERT privilege on the explain tables. If explain tables do not exist, they will be created when you explain the SQL or XQuery statements.
- To view the details of explained statements, including statistics, you will need a minimum of SELECT privilege on both the explain tables and on the system catalog tables.
- To change explained statements, you will need a minimum of UPDATE privilege on the explain tables.
- To remove explained statements, you will need a minimum of DELETE privilege on the explain tables.

To start Visual Explain::

- From the Control Center, right-click a database name and select either **Show Explained Statements History** or **Explain Query**.
- From the Command Editor, execute an explainable statement on the Interactive page or the Script page.
- From the Query Patroller, click **Show Access Plan** from either the Managed Queries Properties notebook or from the Historical Queries Properties notebook.

Troubleshooting Tips:

- Retrieving the access plan when using LONGDATACOMPAT
- Visual Explain up-level and down-level support

Related concepts:

- “Access plan” on page 452
- “Access plan graph” on page 453

Related tasks:

- “Dynamically explaining an SQL or an XQuery statement” on page 464
- “Viewing a graphical representation of an access plan” on page 473
- “Viewing explainable statements for a package” on page 474
- “Viewing the history of previously explained query statements” on page 476

Related reference:

- “Explain tables” on page 466
- “Viewing SQL or XQuery statement details and statistics” on page 469

Visual Explain concepts

This section contains conceptual information specifically related to Visual Explain tasks.

Access plan

Certain data is necessary to resolve an explainable SQL or XQuery statement. An *access plan* specifies an order of operations for accessing this data. An access plan lets you view statistics for selected tables, indexes, or columns; properties for operators; global information such as table space and function statistics; and configuration parameters relevant to optimization. With Visual Explain, you can view the access plan for an SQL or XQuery statement in graphical form.

The optimizer produces an access plan whenever you compile an explainable SQL or XQuery statement. This happens at prep/bind time for static statements, and at run time for dynamic statements.

It is important to understand that an access plan is an *estimate* based on the information that is available. The optimizer bases its estimations on information such as the following:

- Statistics in system catalog tables (if statistics are not current, update them using the RUNSTATS command.)
- Configuration parameters
- Bind options
- The query optimization class

Cost information associated with an access plan is the optimizer's *best estimate* of the resource usage for a query. The actual elapsed time for a query might vary depending on factors outside the scope of DB2 (for example, the number of other applications running at the same time). Actual elapsed time can be measured while running the query, by using performance monitoring.

Related concepts:

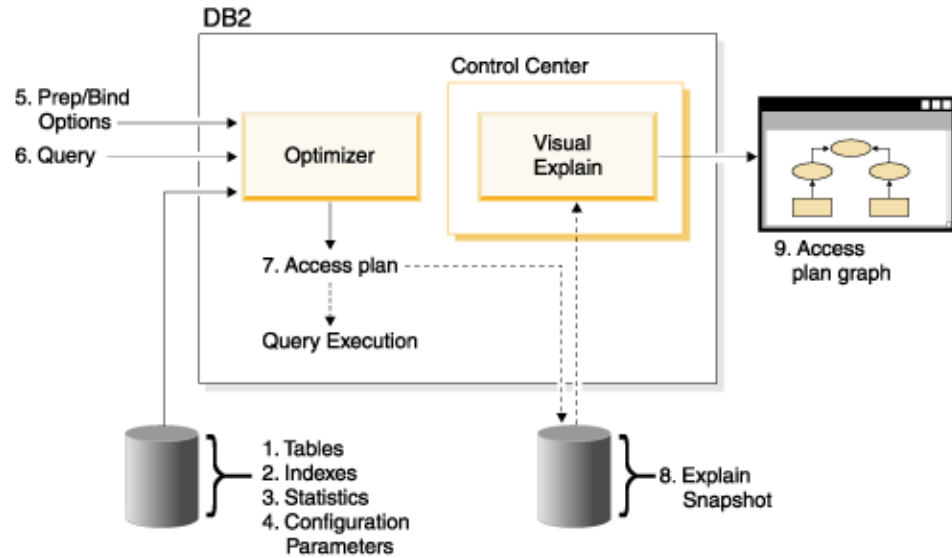
- "Access plan graph" on page 453
- "Visual Explain overview" on page 451

Related tasks:

- "Dynamically explaining an SQL or an XQuery statement" on page 464
- "Viewing a graphical representation of an access plan" on page 473
- "Viewing explainable statements for a package" on page 474
- "Viewing the history of previously explained query statements" on page 476

Access plan graph

Visual Explain uses information from a number of sources in order to produce an access plan graph, as shown in the illustration below. Based on various inputs, the optimizer chooses an access plan, and Visual Explain displays it in an *access plan graph*. The nodes in the graph represent tables and indexes and each operation on them. The links between the nodes represent the flow of data.



Related concepts:

- “Access plan” on page 452
- “Visual Explain overview” on page 451

Related tasks:

- “Viewing the history of previously explained query statements” on page 476
- “Dynamically explaining an SQL or an XQuery statement” on page 464
- “Viewing a graphical representation of an access plan” on page 473
- “Viewing explainable statements for a package” on page 474

Access plan graph node

The access plan graph consists of a tree displaying *nodes*. These nodes represent:

- Tables, shown as rectangles
- Indexes, shown as diamonds
- Operators, shown as octagons (8 sides). TQUEUE operators, shown as parallelograms
- Table functions, shown as hexagons(6 sides).

Related concepts:

- “Access plan” on page 452
- “Access plan graph” on page 453

Clustering

Over time, updates may cause rows on data pages to change location lowering the degree of *clustering* that exists between an index and the data pages. Reorganizing a table with respect to a chosen index reclusters the data. A clustered index is most useful for columns that have range predicates because it allows better sequential access of data in the base table. This results in fewer page fetches, since like values are on the same data page.

In general, only one of the indexes in a table can have a high degree of clustering.

To check the degree of clustering for an index, double-click on its node to display the Index Statistics window. The cluster ratio or cluster factor values are shown in this window. If the value is low, consider reorganizing the table's data.

Related reference:

- "Guidelines for creating indexes" on page 467

Container

A *container* is a physical storage location of the data. It is associated with a table space, and can be a file or a directory or a device.

Related concepts:

- "Table spaces" on page 148

Cost

Cost, in the context of Visual Explain, is the estimated total resource usage necessary to execute the access plan for a statement (or the elements of a statement). Cost is derived from a combination of CPU cost (in number of instructions) and I/O (in numbers of seeks and page transfers).

The unit of cost is the *timeron*. A timeron does not directly equate to any actual elapsed time, but gives a rough relative estimate of the resources (cost) required by the database manager to execute two plans for the same query.

The cost shown in each operator node of an access plan graph is the cumulative cost, from the start of access plan execution up to and including the execution of that particular operator. It does not reflect factors such as the workload on the system or the cost of returning rows of data to the user.

Related concepts:

- "Timerons" in *Administration Guide: Planning*

Dynamic SQL or XQuery

Dynamic SQL or XQuery statements are SQL or XQuery statements that are prepared and executed within an application program while the program is running. In dynamic SQL or XQuery, either:

- You issue the SQL or XQuery statement interactively, using CLI or CLP
- The SQL or XQuery source is contained in host language variables that are embedded in an application program.

When DB2 runs a dynamic SQL or XQuery statement, it creates an access plan that is based on current catalog statistics and configuration parameters. This access plan might change from one execution of the statements application program to the next.

The alternative to dynamic SQL or XQuery is static SQL or XQuery.

Related concepts:

- "Static SQL or XQuery" on page 463

Explain snapshot

With Visual Explain, you can examine the contents of an explain snapshot.

An *explain snapshot* is compressed information that is collected when an SQL statement is explained. It is stored as a binary large object (BLOB) in the EXPLAIN_STATEMENT table, and contains the following information:

- The internal representation of the access plan, including its operators and the tables and indexes accessed
- The decision criteria used by the optimizer, including statistics for database objects and the cumulative cost for each operation.

An explain snapshot is required if you want to display the graphical representation of an SQL statement's access plan. To ensure that an explain snapshot is created:

1. Explain tables must exist in the database manager to store the explain snapshots. For information on how to create these tables, see *Creating explain tables* in the online help.
2. For a package containing static SQL or XQuery statements, set the EXPLSNAP option to ALL or YES when you bind or prep the package. You will get an explain snapshot for each explainable SQL statement in the package. For more information on the **BIND** and **PREP** commands, see the *Command Reference*.
3. For dynamic SQL statements, set the EXPLSNAP option to ALL when you bind the application that issues them, or set the CURRENT EXPLAIN SNAPSHOT special register to YES or EXPLAIN before you issue them interactively. For more information, see the section on current explain snapshots in the *SQL Reference*.

Related tasks:

- “Using explain snapshots” in *DB2 Visual Explain Tutorial*

Related reference:

- “BIND command” in *Command Reference*
- “PRECOMPILE command” in *Command Reference*

Explainable statement

An *explainable statement* is an SQL or XQuery statement for which an explain operation can be performed.

Explainable SQL or XQuery statements are:

- SELECT
- INSERT
- UPDATE
- DELETE
- VALUES

Related concepts:

- “Explained statement” on page 456

Explained statement

An *explained statement* is an SQL or XQuery statement for which an explain operation has been performed. Explained statements are shown in the Explained Statements History window.

Related concepts:

- “Explainable statement” on page 456

Operand

An operand is an entity on which an operation is performed. For example, a table or an index is an operand of various operators such as TBSCAN and IXSCAN.

Related concepts:

- “Operator” on page 457

Operator

An *operator* is either an action that must be performed on data, or the output from a table or an index, when the access plan for an SQL or XQuery statement is executed.

The following operators can appear in the access plan graph:

CMPEXP

Computes expressions. (For debug mode only.)

DELETE

Deletes rows from a table.

EISCAN

Scans a user defined index to produce a reduced stream of rows.

FETCH

Fetches columns from a table using a specific record identifier.

FILTER

Filters data by applying one or more predicates to it.

GENROW

Generates a table of rows.

GRPBY

Groups rows by common values of designated columns or functions, and evaluates set functions.

HSJOIN

Represents a hash join, where two or more tables are hashed on the join columns.

INSERT

Inserts rows into a table.

IXAND

ANDs together the row identifiers (RIDs) from two or more index scans.

IXSCAN

Scans an index of a table with optional start/stop conditions, producing an ordered stream of rows.

MSJOIN

Represents a merge join, where both outer and inner tables must be in join-predicate order.

NLJOIN

Represents a nested loop join that accesses an inner table once for each row of the outer table.

PIPE Transfers rows. (For debug mode only.)

RETURN

Represents the return of data from the query to the user.

RIDSCN

Scans a list of row identifiers (RIDs) obtained from one or more indexes.

RPD (Remote PushDown)

An operator for remote plans. It is very similar to the SHIP operator in Version 8 (RQUERY operator in previous versions), except that it does not contain an SQL or XQuery statement.

SHIP Retrieves data from a remote database source. Used in the federated system.

SORT Sorts rows in the order of specified columns, and optionally eliminates duplicate entries.

TBSCAN

Retrieves rows by reading all required data directly from the data pages.

TEMP Stores data in a temporary table to be read back out (possibly multiple times).

TQUEUE

Transfers table data between database agents.

UNION

Concatenates streams of rows from multiple tables.

UNIQUE

Eliminates rows with duplicate values, for specified columns.

UPDATE

Updates rows in a table.

XISCAN

Scans an index of an XML table.

XSCAN

Navigates an XML document node subtrees.

XANDOR

Allows ANDed and ORed predicates to be applied to multiple XML indexes.

Related concepts:

- “Operand” on page 457

Optimizer

The *optimizer* is the component of the SQL compiler that chooses an access plan for a data manipulation language (DML) SQL statement. It does this by modeling the execution cost of many alternative access plans, and choosing the one with the minimal estimated cost.

Related concepts:

- “Query optimization class” on page 460

Package

A *package* is an object stored in the database that includes the information needed to process the SQL statements associated with one source file of an application program. It is generated by either:

- Precompiling a source file with the **PREP** command
- Binding a bind file that was generated by the precompiler with the **BIND** command.

Related reference:

- “BIND command” in *Command Reference*
- “PRECOMPILE command” in *Command Reference*

Predicate

A *predicate* is an element of a search condition that expresses or implies a comparison operation. Predicates are included in clauses beginning with WHERE or HAVING.

For example, in the following SQL statement:

```
SELECT * FROM SAMPLE
  WHERE NAME = 'SMITH' AND
        DEPT = 895 AND YEARS > 5
```

The following are predicates: NAME = 'SMITH'; DEPT = 895; and YEARS > 5.

Predicates fall into one of the following categories, ordered from most efficient to least efficient:

1. Starting and stopping conditions bracket (narrow down) an index scan. (These conditions are also called range-delimiting predicates.)
2. Index-page (also known as index sargable) predicates can be evaluated from an index because the columns involved in the predicate are part of the index key.
3. Data-page (also known as data sargable) predicates cannot be evaluated from an index, but can be evaluated while rows remain in the buffer.
4. Residual predicates typically require I/O beyond the simple accessing of a base table, and must be applied after data is copied out of the buffer page. They include predicates that contain subqueries, or those that read LONG VARCHAR or LOB data stored in files separate from the table.

When designing predicates, you should aim for the highest selectivity possible so that the fewest rows are returned.

The following types of predicates are the most effective and the most commonly used:

- A *simple equality join predicate* is required for a merge join. It is of the form table1.column = table2.column, and allows columns in two different tables to be equated so that the tables can be joined.
- A *local predicate* is applied to one table only.

Related concepts:

- “Selectivity of predicates” on page 461

Query optimization class

A *query optimization class* is a set of query rewrite rules and optimization techniques for compiling queries.

The primary query optimization classes are:

- 1 Restricted optimization. Useful when memory and processing resources are severely restrained. Roughly equivalent to the optimization provided by Version 1.
- 2 Slight optimization. Specifies a level of optimization higher than that of Version 1, but at significantly less optimization cost than levels 3 and above, especially for very complex queries.
- 3 Moderate optimization. Comes closest to matching the query optimization characteristics of DB2 for MVS/ESA.
- 5 Normal optimization. Recommended for a mixed environment using both simple transactions and complex queries.
- 7 Normal optimization. The same as query optimization 5 except that it does not reduce the amount of query optimization for complex dynamic queries.

Other query optimization classes, to be used only under special circumstances, are:

- 0 Minimal optimization. Use only when little or no optimization is required (that is, for very simple queries on well-indexed tables).
- 9 Maximum optimization. Uses substantial memory and processing resources. Use only if class 5 is insufficient (that is, for very complex and long-running queries that do not perform well at class 5).

In general, use a higher optimization class for static queries and for queries that you anticipate will take a long time to execute, and a lower optimization class for simple queries that are submitted dynamically or that are run only a few times.

To set the query optimization for dynamic SQL or XQuery statements, enter the following command in the command line processor:

```
SET CURRENT QUERY OPTIMIZATION = n;
```

where 'n' is the desired query optimization class.

To set the query optimization for static SQL or XQuery statements, use the QUERYOPT option on the **BIND** or **PREP** commands.

Related concepts:

- "Optimizer" on page 458

Related reference:

- "BIND command" in *Command Reference*
- "PRECOMPILE command" in *Command Reference*

Cursor blocking

Cursor blocking is a technique that reduces overhead by having the database manager retrieve a *block* of rows in a single operation. These rows are stored in a cache while they are processed. The cache is allocated when an application issues

an OPEN CURSOR request, and is de-allocated when the cursor is closed. When all the rows have been processed, another block of rows is retrieved.

Use the BLOCKING option on the PREP or BIND commands along with the following parameters to specify the type of cursor blocking:

UNAMBIG

Only unambiguous cursors are blocked (the default).

ALL Both ambiguous and unambiguous cursors are blocked.

NO Cursors are not blocked.

Related tasks:

- “Specifying row blocking to reduce overhead” in *Performance Guide*

Related reference:

- “BIND command” in *Command Reference*
- “PRECOMPILE command” in *Command Reference*

Selectivity of predicates

Selectivity refers to the probability that any row will satisfy a predicate (that is, be true).

For example, a selectivity of 0.01 (1%) for a predicate operating on a table with 1,000,000 rows means that the predicate returns an estimated 10,000 rows (1% of 1,000,000), and discards an estimated 990,000 rows.

A highly selective predicate (one with a selectivity of 0.10 or less) is desirable. Such predicates return fewer rows for future operators to work on, thereby requiring less CPU and I/O to satisfy the query.

Example

Suppose that you have a table of 1,000,000 rows, and that the original query contains an ‘ORDER BY’ clause requiring an additional sorting step. With a predicate that has a selectivity of 0.01, the sort would have to be done on an estimated 10,000 rows. However, with a less selective predicate of 0.50, the sort would have to be done on an estimated 500,000 rows, thus requiring more CPU and I/O time.

Related concepts:

- “Predicate” on page 459

Sequences

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.

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 tasks:

- “Creating a sequence” on page 234

Star join

A set of joins are considered to be a star join when a fact table (large central table) is joined to two or more dimension tables (smaller tables containing descriptions of the column values in the fact table).

A Star join is comprised of 3 main parts:

- Semijoins
- Index ANDing of the results of the Semijoins
- Completing the semijoins.

It shows up as two or more joins feeding an IXAND operator.

A Semijoin is a special form of join in which the result of the join is only the Row Identifier (RID) of the inner table, instead of the joining of the inner and outer table columns.

Star joins use Semijoins to supply Row Identifiers to an Index ANDing operator. The Index ANDing operator accumulates the filtering affect of the various joins. The output from the Index ANDing operator is fed into an Index ORing operator, which orders the Row Identifiers, and eliminates any duplicate rows that may have

resulted from the joins feeding the Index ANDing operator. The rows from the fact table are then fetched, using a Fetch operator. Finally, the reduced fact table is joined to all of the dimension tables, to complete the joins.

Performance suggestions:

- Create indexes on the fact table for each of the dimension table joins.
- Ensure the sort heap threshold is high enough to allow allocating the Index ANDing operator's bit filter. For star joins, this could require as much as 12MB, or 3000 4K pages. For Intra-partition parallelism, the bit filter is allocated from the same shared memory segment as the shared sort heap, and it is bounded by the *sortheap* database configuration parameter and the *sheapthres_shr* database configuration parameter.
- Apply filtering predicates against the dimension tables. If statistics are not current, update them using the runstats command.

Related reference:

- "IXAND operator" in *Performance Guide*
- "RUNSTATS command" in *Command Reference*
- "Using RUNSTATS" on page 468

Static SQL or XQuery

A *static SQL or XQuery* statement is embedded within an application program. All these embedded statements must be precompiled and bound into a *package* before the application can be executed. To execute XQuery expressions in static SQL, use the XMLQUERY function.

When DB2 compiles these statements, it creates an access plan for each one that is based on the catalog statistics and configuration parameters at the time that the statements were precompiled and bound.

These access plans are always used when the application is run; they do not change until the package is bound again.

The alternative to static SQL or XQuery is dynamic SQL or XQuery.

Related tasks:

- "Executing XQuery expressions in embedded SQL applications" in *Developing Embedded SQL Applications*

Visual Explain

Note: As of Version 6, Visual Explain can no longer be invoked from the command line. It can still, however, be invoked from various database objects in the Control Center. For this version, the documentation continues to use the name Visual Explain.

Visual Explain lets you view the access plan for explained SQL or XQuery statements as a graph. You can use the information available from the graph to tune your queries for better performance.

An access plan graph shows details of:

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

You can also use Visual Explain to:

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

To start Visual Explain::

- From the Control Center, right-click a database name and select either **Show Explained Statements History** or **Explain Query**.
- From the Command Editor, execute an explainable statement on the Interactive page or the Script page.
- From the Query Patroller, click **Show Access Plan** from either the Managed Queries Properties notebook or from the Historical Queries Properties notebook.

Related concepts:

- “Visual Explain overview” on page 451

Dynamically explaining an SQL or an XQuery statement

Use the Explain Query Statement window to dynamically explain an SQL or XQuery statement and to produce an access plan graph. If explain tables do not exist, they will be created.

An explained statement record is added to the Explained Statements History for all successful operations.

Prerequisites:

To dynamically explain query statements, you will need at least the INSERT privilege on the explain tables.

Procedure:

To dynamically explain an SQL or XQuery statement:

1. Open the Explain Query Statement window: From the Control Center, expand the object tree until you find the **Databases** folder, expand the **Databases** folder until you find the database that you want, and then do one of the following:
 - Right-click the database, and click **Explain Query** from the pop-up menu. Highlight a database, and click **Selected->Explain Query**. The Explain Query Statement window opens.
 - Open the Explainable Statements, the Explained Statements History, or the Access Plan Graph window. Select **Statement->Explain Query**. The Explain Query Statement window opens.

Note:

- If you select **Explain Query** from the Control Center, the **Query text** field will be empty.

- If you selected **Explain Query** from the Explainable Statements or Explained Statements History windows and selected an entry in that window, the **Query text** field will be populated with the SQL or XQuery statement related to that entry. If you did not select an entry, the **Query text** field will be empty.
 - If you selected **Explain Query** from the Access Plan Graph window, the **Query text** field will contain text for the SQL or XQuery statement whose access plan was shown in the graph.
2. In the **Query text** field, you can:
 - Type an SQL or XQuery statement that you want explained.
 - Change the text of an SQL or XQuery statement that already appears in the text field.
 - Get an SQL or XQuery statement from a specified file.
 - Save the SQL or XQuery statement to a specified file.
 3. Optional: In the **Query number** or **Query tag** fields, type new values.
 4. Optional: In the **Query optimization class** field, type new values.
 5. Optional: Select the **Populate all columns in explain tables** check box to populate all of the columns of the explain tables from the dynamic explain; otherwise, only the few columns needed by Visual Explain will be populated from the dynamic explain.

Related concepts:

- “Visual Explain overview” on page 451

Related tasks:

- “Viewing explainable statements for a package” on page 474
- “Viewing the history of previously explained query statements” on page 476
- “Viewing a graphical representation of an access plan” on page 473

Related reference:

- “Viewing SQL or XQuery statement details and statistics” on page 469




Creating an access plan using the Command Editor

Use the Command Editor to generate, edit, execute, and manipulate SQL and XQuery statements, IMS commands, and DB2 commands. You can also use the Command Editor to work with the resulting output and to view a graphical representation of the access plan for explained SQL statements. You can execute commands and SQL statements on DB2 databases for Linux and Windows, for z/OS and OS/390 systems and subsystems, and for IMSplexes.

Procedure:

To create an access plan using the Command Editor:

1. Open the Command Editor: To open a stand-alone Command Editor, select **Start -> Programs -> IBM DB2 -> Command Line Tools -> Command Editor**. For other methods, see the Command Editor overview.
2. Select either the **Interactive** or **Script** tab, and do the following:
 - a. Connect to a database. (Type the connect command in the text area and select **Execute** from the Interactive or Script menu, depending on which

- page you selected in step 2., or click on the  icon, or press the **Ctrl+Enter** keys to execute the command.)
- b. To create an access plan without executing the statement, type an explainable statement in the text area and select **Create access plan**, from the Interactive or Script menu, or click on the  icon. The access plan graph is displayed on the Access Plan page.
You can also select an explainable statement from an existing script.
 - c. To create an access plan and also execute the statement:
 - 1) Select **Options** from the Interactive or Script menu. The Command Center Options notebook opens. Click on the Access Plan tab. Select the **Automatically generate access plan** check box, and click **OK**.
 - 2) Type an *explainable statement* in the text area or select an existing statement. Select **Execute**, from the Interactive or Script menu, or click the  icon. The results are displayed on the Results page. To view the generated access plan, click on the Access Plan tab.

Related concepts:

- “Command Editor overview” in *Online DB2 Information Center*
- “Explainable statement” on page 456
- “Access plan” on page 452
- “Visual Explain overview” on page 451

Related tasks:

- “Executing commands and SQL statements using the Command Editor” in *Online DB2 Information Center*

Explain tables

To create explain snapshots, you must ensure that the following explain tables exist for your user ID:

- EXPLAIN_INSTANCE
- EXPLAIN_STATEMENT

To check if they exist, use the **DB2 list tables** command. If you would like to use Visual Explain, and these tables do not exist, you must create them using the following instructions:

1. If DB2 has not already been started, issue the **db2start** command.
2. From the DB2 CLP prompt, connect to the database that you want to use. To connect to the SAMPLE database, issue the **connect to sample** command.
3. Create the explain tables, using the sample command file that is provided in the EXPLAIN.DDL file. This file is located in the sqllib\misc directory. To run the command file, go to this directory and issue the **db2 -tf EXPLAIN.DDL** command. This command file creates explain tables that are prefixed with the connected user ID. This user ID must have CREATETAB privilege on the database, or SYSADM or DBADM authority.

Note: Before you run **db2 -tvf EXPLAIN.DDL**, ensure that explain tables for the schema name do not exist. If you have migrated from an earlier version, you need to run **db2exmig** to migrate the explain tables.

Related concepts:

- “Visual Explain overview” on page 451

Related tasks:

- “Viewing the history of previously explained query statements” on page 476
- “Dynamically explaining an SQL or an XQuery statement” on page 464
- “Viewing explainable statements for a package” on page 474

Guidelines for creating indexes

Creating appropriate indexes allows the optimizer to choose an index scan for those cases where it would be more efficient than a table scan.

Some guidelines for creating indexes include:

- Define primary keys and unique indexes wherever they apply.
- Create an index on any column that the query uses to join tables (join predicates).
- Create an index on any column from which you search for particular values on a regular basis.
- Create an index on columns that are commonly used in ORDER BY clauses.
- Ensure that you have used predicates that retrieve only the data you need. For example, ensure that the selectivity value for the predicates represents the portion of the table that you want returned.
- When creating a multicolumn index, the first columns of the index should be the ones that are used most often by the predicates in your query.
- Ensure that the disk and update maintenance overhead an index introduces will not be too high.

Related concepts:

- “Space requirements for indexes” in *Administration Guide: Planning*
- “Visual Explain overview” on page 451

Related tasks:

- “Estimating space requirements for tables and indexes” on page 272

Out-of-date access plans

Symptom

The STATS_TIME row indicates that the statistics are not updated.

Possible cause

The optimizer used default values. (These default values are displayed with the keyword “default”.) This situation can result in an out-of-date access plan.

Action

It is recommended that you use the runstats command to update the statistics; then rebind the package.

Related concepts:

- “Access plan” on page 452
- “Visual Explain overview” on page 451

Retrieving the access plan when using LONGDATACOMPAT

Symptom:

No explained statement history or access plan can be displayed using Visual Explain.

Possible cause:

If the value for LONGDATACOMPAT is set to 1 in the db2cli.ini file, the Visual Explain access plan can be generated but cannot be retrieved.

Action:

As a work around, a database alias can be created for that database with LONGDATACOMPAT set to 0. For example:

```
DB2 UPDATE CLI CFG FOR SECTION db-alias-name USING LONGDATACOMPAT 0
```

To check the CLI configuration values, the following command can be used:

```
GET CLI CONFIGURATION [AT GLOBAL LEVEL] [FOR SECTION section-name]
```

For instance, if the database name is called sample:

```
GET CLI CONFIGURATION FOR SECTION sample
```

Related concepts:

- “Access plan” on page 452

Using RUNSTATS

The optimizer uses the catalog tables from a database to obtain information about the database, the amount of data in it, and other characteristics, and uses this information to choose the best way to access the data. If current statistics are not available, the optimizer might choose an inefficient access plan based on inaccurate default statistics.

It is highly recommended that you use the **RUNSTATS** command to collect current statistics on tables and indexes, especially if significant update activity has occurred or new indexes have been created since the last time the **RUNSTATS** command was executed. This provides the optimizer with the most accurate information with which to determine the best access plan.

Be sure to use **RUNSTATS** *after* making your table updates; otherwise, the table might appear to the optimizer to be empty. This problem is evident if cardinality on the Operator Details window equals zero. In this case, complete your table updates, rerun the **RUNSTATS** command and recreate the explain snapshots for affected tables.

Note:

- Use **RUNSTATS** on all tables and indexes that might be accessed by a query.
- The quantile and frequent value statistics determine when data is unevenly distributed. To update these values, use **RUNSTATS** on a table with the **WITH DISTRIBUTION** clause.
- In addition to statistics, other factors (such as the ordering of qualifying rows, table size, and buffer pool size) might influence how an access plan is selected.

- Applications should be rebound (and their statements optionally re-explained) after you run the **RUNSTATS** command or change configuration parameters.

The **RUNSTATS** command (which can be entered from the DB2 CLP prompt) can provide different levels of statistics as shown in the following syntax:

Basic Statistics

Table:

RUNSTATS ON TABLE tablename

Index:

RUNSTATS ON TABLE tablename FOR INDEXES ALL

Both tables and indexes:

RUNSTATS ON TABLE tablename AND INDEXES ALL

Enhanced Statistics

Table:

RUNSTATS ON TABLE tablename WITH DISTRIBUTION

Index:

RUNSTATS ON TABLE tablename FOR DETAILED INDEXES ALL

Both tables and indexes:

RUNSTATS ON TABLE tablename WITH DISTRIBUTION AND
DETAILED INDEXES ALL

Note: In each of the above commands, the tablename *must* be fully qualified with the schema name.

Related concepts:

- “Visual Explain overview” on page 451

Related reference:

- “RUNSTATS command” in *Command Reference*

Viewing SQL or XQuery statement details and statistics

Use Visual Explain to view details and statistics for SQL or XQuery statements.

To start Visual Explain::

- From the Control Center, right-click a database name and select either **Show Explained Statements History** or **Explain Query**.
- From the Command Editor, execute an explainable statement on the Interactive page or the Script page.
- From the Query Patroller, click **Show Access Plan** from either the Managed Queries Properties notebook or from the Historical Queries Properties notebook.

For more information on the windows described below, refer to the online help.

Table 24. Viewing SQL or XQuery statement text...

Tasks	Procedure
To view the text for an SQL or XQuery statement:	Use the Query Text window. To open this window, open either the Explainable Statements or the Explained Statements History window, or the Access Plan Graph window. Select Statement → Show Query Text .
To view the text for an explained SQL or XQuery statement that was rewritten by the optimizer:	Use the Optimized Query Text window. To open this window, open the Access Plan Graph window. Select Statement → Show Optimized Query Text .
To find a specific character string in the text of the window that you are using:	Use the Find window. To open this window, open either the Query Text or the Optimized Query Text window and click the Find push button.

Table 25. Viewing SQL or XQuery statement details...

Tasks	Procedure
To view the list of built-in functions and user-defined functions that are associated with the SQL or XQuery statement whose access plan is shown in the graph:	Use the Functions window. To open this window, open the Access Plan Graph window. Select Statement → Show Statistics → Functions .
To view the list of indexes that are defined on the table that is shown in the Table Statistics window:	Use the Indexes window. To open this window, open the Table Statistics window and click the Indexes push button.
to view details for an operator that is selected in the Access Plan Graph window:	Use the Operator details window. To open this window, open the Access Plan Graph window. To see statistics on an operator node in the graph, do one of the following: <ul style="list-style-type: none"> • Highlight the operator node and select Node→Show Details. • Double-click the operator node. • Right-click the operator node and select Show Details from the pop-up menu.
To view the configuration parameters and bind options that affect the optimization process:	Use the Optimization Parameters window. Current values are shown, as well as values from the time of the explain. To open this window, open the Access Plan Graph window. Select Statement → Show Optimization Parameters .
To view a list of the column groups that are associated with the SQL or XQuery statement whose access plan is shown in the graph:	Use the Column Groups window. These columns belong to the table that is shown in the Table Statistics window. To open this window, open the Table Spaces window. Select one or more entries, and click OK .

Table 25. Viewing SQL or XQuery statement details... (continued)

Tasks	Procedure
To view a list of the referenced columns that are associated with the SQL or XQuery statement whose access plan is shown in the graph:	Use the Referenced Columns window. These columns belong to the table that is shown in the Table Statistics window. To open this window, open the Table Statistics window and click the Referenced Columns push button.
To view statistics for a column group that is referenced in a selected table:	Use the Referenced Column Groups window. To open this window, open the Referenced Columns window. Select one or more entries, and click on OK .
To view the list of table spaces that are associated with the SQL or XQuery statement whose access plan is shown in the graph:	Use the Table Spaces window. To open this window, open the Access Plan Graph window. Select Statement → Show Statistics → Table Spaces .

Table 26. Viewing SQL or XQuery statement statistics...

Tasks	Procedure
To view column distribution values for the column that is shown in the Referenced Column Statistics window:	Use the Column Distribution Statistics window. To open this window, open the Referenced Column Statistics window and click the Column Distribution push button.
To view statistics for a built-in or user defined function that is associated with the explained SQL or XQuery statement:	Use the Function Statistics window. To open this window, open the Functions window. Select one or more entries, and click OK .
To view statistics for an index node selected in the Access Plan Graph or an index entry selected in the Indexes window:	Use the Index Statistics window. To open this window, open the Access Plan Graph window. Do one of the following: <ul style="list-style-type: none"> • To see statistics on an Index node in the graph, do one of the following: <ul style="list-style-type: none"> – Highlight the index node and select Node→Show Statistics. – Double-click the index node. – Right-click the index node and select Show Statistics from the pop-up menu. • To see statistics on other indexes defined for a Table node in the graph, do one of the following: <ul style="list-style-type: none"> – Highlight the table node and select Node→Show Statistics. – Double-click the table node. – Right-click the table node and select Show Statistics from the pop-up menu. <p>The Table Statistics window opens. click the Indexes push button to open the Indexes window. Select one or more entries in the Indexes window and click OK. An Index Statistics window opens for each entry that you select.</p>

Table 26. Viewing SQL or XQuery statement statistics... (continued)

Tasks	Procedure
To view the estimated number of page fetches for each hypothetical number of buffer pages as an ordered set of pairs:	Use the Page Fetch Pairs window. The numbers model the number of I/Os required to read the data pages into buffer pools of various sizes. To open this window, open the Index Statistics window and click the Page Fetch Pairs push button.
To view statistics for a column that is referenced in a selected table:	Use the Referenced Column Statistics window. To open this window, open the Referenced Columns window. Select one or more entries, and click OK .
To view statistics for a table function node selected in the access plan graph:	Use the Table Function Statistics window. To open this window, open the Access Plan Graph window. To see statistics on a Table function node in the graph, do one of the following: <ul style="list-style-type: none"> • Highlight the table function node and select Node→Show Statistics. • Double-click the table function node. • Right-click the table function node and select Show Statistics from the pop-up menu. <p>Note: If the node is GENROW, only explained statistics are displayed; otherwise, both explained and current statistics are displayed.</p>
To view statistics for the table node selected in the Access Plan Graph:	Use the Table Statistics window. To open this window, open the Access Plan Graph window. To see statistics on a Table node in the graph, do one of the following: <ul style="list-style-type: none"> • Highlight the table node and select Node→Show Statistics. • Double-click the table node. • Right-click the table node and select Show Statistics from the pop-up menu.
To view statistics for a table space that is associated with an explained SQL or XQuery statement:	Use the Table Space Statistics window. To open this window, open the Table Spaces window. Select one or more entries, and click OK .

Note on CARD row under the Statistics column: In a partitioned database environment, the value in the **Current** column is computed based on all nodes, while the value in the **Explained** column is computed based on a particular node.

Related concepts:

- “Visual Explain overview” on page 451

Related tasks:

- “Viewing a graphical representation of an access plan” on page 473
- “Viewing explainable statements for a package” on page 474

- “Viewing the history of previously explained query statements” on page 476


Viewing a graphical representation of an access plan

Use the Access Plan Graph window to view a graphical representation of the access plan of an explained SQL or XQuery statement. The nodes in the graph represent tables and indexes and each operation on them. The links between the nodes represent the flow of data.

Tasks:

- Use the **Statement** menu to print the graph, to dynamically explain an SQL or XQuery statement, to view the text or optimized text, or to view optimization parameters or statistics.
- Use the **Node** menu to view details or statistics on the nodes, or to get additional help on each of the operators.
- Use the **View** menu to change the graph settings or to see an overview of the graph. This is particularly useful for large graphs.

From this window, you can view details about the following objects:

- Table spaces and table space statistics
- Functions and function statistics
- Operators
-  Partitioned databases
- Operands
 - Column distribution statistics
 - Index and index statistics
 - Page fetch pairs statistics
 - Column groups
 - Referenced columns, referenced column groups, and referenced column statistics
 - Table function statistics and table statistics

To open the Access Plan Graph window, use one of the following methods:

1. Open either the Explainable Statements, or the Explained Statements History window. Select **Statement**→**Show Access Plan**. The Access Plan Graph window opens.
2. Invoke **Explain Query** from either the Explainable Statements or the Explained Statements History window. The Explain Query statement window opens as a result of the dynamic explain.





Reading the contents of the Access Plan Graph window:

Top area of the window

The top area of the Access Plan Graph window identifies the statement whose access plan is displayed on the graph.

This part of the window also shows:

- The statement’s explain date, time, package name, and version.
- If the Federated function was enabled at the time the statement was created.
- Its total estimated cost

- The type of parallelism of the system in which this statement is explained. It can be one of the following types:
 -  None
 -  Intra-partition parallelism
 -  Inter-partition parallelism
 -  Full parallelism (intra-partition and inter-partition)

The graph

The nodes in the graph represent operands (tables, indexes, or table functions), and the operators that act on them. To view detailed statistical information for a node, double-click on it.

To view the information shown in the graph in more detail, drag the zoom slider up or down.

Float values might be presented in scientific notation.

Troubleshooting Tips:

- Retrieving the access plan when using LONGDATACOMPAT
- Visual Explain up-level and down-level support

Related concepts:

- “Access plan” on page 452
- “Access plan graph node” on page 454
- “Operator” on page 457
- “Operand” on page 457
- “Cost” on page 455
- “Visual Explain overview” on page 451

Related tasks:

- “Viewing explainable statements for a package” on page 474
- “Viewing the history of previously explained query statements” on page 476

Related reference:

- “Retrieving the access plan when using LONGDATACOMPAT” on page 468
- “Visual Explain support for earlier and later releases” on page 478
- “Viewing SQL or XQuery statement details and statistics” on page 469

Viewing explainable statements for a package

Use the Explainable Statements window to view the explainable query statements for a selected package.

If an explain snapshot has been taken for a statement, you can use this list to view additional information about that statement (such as its total cost and a graphical view of its access plan).

Tasks:

- Use the **Statement** menu to view the history of previously explained SQL or XQuery statements, to view a graphical representation of the access plan, to dynamically explain a query statement, and to view text for a query statement.

- Use the **View** menu, or the icons on the secondary toolbar to sort, filter, or customize the explainable statements. You can also save the contents of this window using the options in this menu.

To open the Explainable Statements window, do the following:

- From the Control Center, expand the object tree until you find the **Packages** folder (under the **Application Objects** folder).
- click the **Packages** folder. Any existing package objects are displayed in the pane on the right side of the window.
- Do one of the following:
 - Right-click the package you want and select **Show Explainable Statements** from the pop-up menu.
 - Highlight the package and select **Selected->Show Explainable Statements**.
 - Double-click the package.

Reading the contents of the Explainable Statements window:

The columns in the window provide the following information about SQL or XQuery statements:

Statement number

The line number of the SQL or XQuery statement in the source module of the application program. For static queries, this number corresponds to the STMTNO column in the SYSCAT.STATEMENTS table.

Section number

The number of the section within the package that is associated with the SQL or XQuery statement.

Explain snapshot

States whether an explain snapshot has been taken for the SQL or XQuery statement. (If it has not been taken, you cannot view an access plan graph for the statement.)

Total cost

The estimated total cost (in timerons) of returning the query results for the selected SQL or XQuery statement. (Available only if the package containing the statement has been explained previously.)

Query text

The first 100 characters of the query statement. (Use the scroll bar at the bottom of the window to scroll through it.) To view the complete SQL or XQuery statement, select **Statement->Show Query Text**.

Troubleshooting Tips:

- Retrieving the access plan when using LONGDATACOMPAT
- Visual Explain up-level and down-level support

Related concepts:

- “Package” on page 459
- “Explain snapshot” on page 456
- “Cost” on page 455
- “Access plan” on page 452
- “Visual Explain overview” on page 451

Related tasks:

- “Viewing a graphical representation of an access plan” on page 473
- “Viewing the history of previously explained query statements” on page 476

Related reference:

- “Retrieving the access plan when using LONGDATACOMPAT” on page 468
- “Visual Explain support for earlier and later releases” on page 478
- “Viewing SQL or XQuery statement details and statistics” on page 469

Viewing the history of previously explained query statements

Use the Explained Statements History window to view the history of previously explained SQL or XQuery statements for a selected database. Each entry is an explained statement that is associated with either:

- A static SQL or XQuery statement in a package
- A dynamic SQL or XQuery statement.

Tasks:

- Use the **Statement** menu to view a graphical representation of an access plan, to dynamically explain a query statement, to view text for a query statement, or to change or remove a query statement.
- Use the **View** menu, or the icons on the secondary toolbar to sort, filter, or customize the explainable statements. You can also save the contents of this window using the options in this menu.

To open Explained Statements History window, do one of the following:

- From the Control Center, expand the object tree until you find the **Databases** folder, expand the folder until you find the database you want, and then do one of the following:
 - Right-click the database and select **Show Explained Statements History** from the pop-up menu. or select **Selected->Show Explained Statements History**.
 - Highlight the database and select **Selected->Show Explained Statements History**.
- From the Control Center, expand the object tree until you find the **Packages** folder (under the **Application Objects** folder). Then:
 - click the **Packages** folder. Any existing package objects are displayed on the right side of the window.
 - Right-click the package that you want, and select **Show Explained Statements History** from the pop-up menu; or highlight the package and select **Selected->Show Explained Statements History**; or simply double-click the package.
- From the Explainable Statements window, select **Statement->Show Explained Statements History**.

If a statement is selected in the Explainable Statements window, the Explained Statements History window shows all of the explained statements that are related to the selected SQL statements.

If no statement is selected, the Explained Statements History window shows all the explained statements that are related to the package that the explainable statements are in.

The Explained Statements History window may or may not contain explained statements, depending on whether the explain tables exist.

Reading the contents of the Explained Statements History window:

The columns in the window provide the following information about the query statements that have been explained:

Package name

The name of the package that either:

- Contains the SQL or XQuery statement (in the case of a static query)
- Issued the SQL or XQuery statement (in the case of a dynamic query).

Package creator

The user ID of the user who created the package.

Package version

The version number of the package.

Explain snapshot

States whether an explain snapshot has been taken for the SQL or XQuery statement. (If it has not, you cannot view an access plan graph for the statement.)

Latest bind

If the statement is contained in a package, this field indicates whether or not the statement is associated with the latest bound package.

Dynamic explain

States whether the explained query statement was dynamic. (If it was not, it was a Static SQL or XQuery statement in a package.)

Explain date

The date when the statement had an explain operation performed on it.

Explain time

The time when the statement had an explain operation performed on it.

Total cost

The estimated total cost (in timerons) of the statement.

Statement number

The line number of the SQL or XQuery statement in the source module of the application program.

Section number

The number of the section within the package that is associated with the SQL or XQuery statement.

Query number

The query number that is associated with the statement.

Query tag

The query tag that is associated with the statement.

Query text

The first 100 characters of the original SQL or XQuery statement. (Use the scroll bar at the bottom of the window to scroll through it.) To view the complete SQL or XQuery statement, select Statement->Show Query Text.

Remarks

Any remarks associated with the statement. (For example, for a static query statement, the remark associated with the package containing the statement.)

Troubleshooting Tips:

- Retrieving the access plan when using LONGDATACOMPAT
- Visual Explain up-level and down-level support

Related concepts:

- “Explained statement” on page 456
- “Package” on page 459
- “Explain snapshot” on page 456
- “Dynamic SQL or XQuery” on page 455
- “Static SQL or XQuery” on page 463
- “Cost” on page 455
- “Visual Explain overview” on page 451

Related tasks:

- “Viewing a graphical representation of an access plan” on page 473
- “Viewing explainable statements for a package” on page 474

Related reference:

- “Retrieving the access plan when using LONGDATACOMPAT” on page 468
- “Visual Explain support for earlier and later releases” on page 478
- “Viewing SQL or XQuery statement details and statistics” on page 469

Visual Explain support for earlier and later releases

Earlier releases: supported

However, if you are running Visual Explain on a Version 9 client accessing a Version 8 database, Visual Explain does handle the Version 8 snapshots. Visual Explain supports earlier release compatibility.

Later releases: not supported

When running Visual Explain on a Version 8 client accessing a Version 9 database, Visual Explain returns an error when it tries to parse the Version 9 data. Visual Explain does not support this upward level compatibility since the snapshots generated by Version 9 are different from those generated by Version 8.

Related concepts:

- “Visual Explain” on page 463

Part 2. Database Security

Chapter 8. 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 9, “Auditing DB2 database activities,” on page 621.

The following topics are discussed:

- “Security issues when installing the DB2 database manager”
- “Authentication methods for your server” on page 490
- “Authentication considerations for remote clients” on page 495
- “Partitioned database authentication considerations” on page 496
- “Introduction to firewall support” on page 619
- “Authorization, privileges, and object ownership” on page 501
- “Controlling access to database objects” on page 519
- “Tasks and required authorizations” on page 608
- “Using the system catalog for security issues” on page 609.

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 the DB2 database manager

Security considerations are important to the DB2 administrator from the moment the product is installed.

To complete the installation of the DB2 database manager, a user ID, a group name, and a password are required. The GUI-based DB2 database manager install program creates default values for different user IDs and the group. Different defaults are created, depending on whether you are installing on UNIX or Windows platforms:

- On UNIX and Linux platforms, if you choose to create a DB2 instance in the instance setup window, the DB2 database install program creates, by default, different users for the DAS (dasusr), the instance owner (db2inst), and the fenced user (db2fenc). Optionally, you can specify different user names

The DB2 database 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 database 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 database 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 database install program creates, by default, the user db2admin for the DAS user, the instance owner, and fenced users (you can specify a different user name during setup, if you want). 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.

When working with DB2 Data Partitioning Feature (DPF) on UNIX operating system environments, the DB2 database manager by default uses the rsh utility (remsh on HP-UX) to run some commands on remote nodes. The rsh utility transmits passwords in clear text over the network, which can be a security exposure if the DB2 server is not on a secure network. You can use the DB2RSHCMD registry variable to set the remote shell program to a more secure alternative that avoids this exposure. One example of a more secure alternative is ssh. See the DB2RSHCMD registry variable documentation for restrictions on remote shell configurations.

After installing the DB2 database manager, 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 environments	A valid DB2 database user name that belongs to the Administrators group.
UNIX platforms	A valid DB2 database user name that belongs to the primary group of the instance owner.

SYSADM privileges are the most powerful set of privileges available within the DB2 database manager. As a result, you might not want all of these users to have SYSADM privileges by default. The DB2 database manager 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 the DB2 database 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.

Note: To avoid accidentally deleting or overwriting instance configuration or other files, administrators should consider using another user account, which does not belong to the same primary group as the instance owner, for day-to-day administration tasks that are performed on the server directly.

Related concepts:

- "General naming rules" on page 663
- "User, user ID and group naming rules" on page 666
- "Authentication" in *Administration Guide: Planning*
- "Authorization" in *Administration Guide: Planning*
- "Naming rules in a Unicode environment" on page 669
- "Naming rules in an NLS environment" on page 668
- "Location of the instance directory" on page 489
- "UNIX platform security considerations for users" on page 489
- "Windows platform security considerations for users" on page 485

Related reference:

- "Communications variables" in *Performance Guide*

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 database server.
- `TOKENLOCAL`
This choice enables access token support to lookup all local groups that the user belongs to on the DB2 database 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, the DB2 database system 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 the DB2 database system, 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 the DB2 database system should look up 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 database 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 queries 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.

Related concepts:

- “Security issues when installing the DB2 database manager” on page 481

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.

Windows platform security considerations for users

System Administration (SYSADM) authority is granted to any valid DB2 database 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 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, DB2 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 489

Windows local system account support

On Windows platforms (except Windows ME), the DB2 database system 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 database has restrictions on objects with schema names starting with "SYS". Therefore if your applications contain DDLs that create DB2 database objects, they should be written such that:

- For static queries, they should be bounded with a value for the QUALIFIER options other than the default one.
- For dynamic queries, the objects to be created should be explicitly qualified with a schema name supported by the DB2 database, or the CURRENT SCHEMA register must be set to a schema name supported by the DB2 database.

Group information for the LSA is gathered at the first group lookup request after the DB2 database instance is started and will not be refreshed until the instance is restarted.

Note: Applications running under the context of the local system account (LSA) are supported on all Windows platforms, except Windows ME.

Related concepts:

- "Security issues when installing the DB2 database manager" on page 481

Extended Windows security using DB2ADMNS and DB2USERS groups

For the server version of the DB2 database manager, extended security is implicitly *enabled* by default. However, for the client version, extended security is implicitly *disabled* by default; you *must* explicitly select extended security during installation to have it enabled. During DB2 installation on a client, select the Enable operating system security check box on the Enable operating system security for DB2 object panel. the installer creates two new groups, DB2ADMNS and DB2USERS. DB2ADMNS and DB2USERS are the default group names; optionally, you can specify different names for these groups at installation time (if you select silent install, you can change these names within the install response file). If you choose to use groups that already exist on your system, be aware that the privileges of these groups will be modified. They will be given the privileges, as required, listed in the table, below. It is important to understand that these groups are used for protection at the *operating-system* level and are in no way associated with DB2 authority levels, such as SYSADM, SYSMANT, and SYSCTRL. However, instead of using the default Administrator's group, your database administrator can use the DB2ADMNS group for one or all of the DB2 authority levels, at the discretion of the installer or administrator. It is recommended that if you are specifying a SYSADM group, then that should be the DB2ADMNS group. This can be established during installation or subsequently, by an administrator.

The DB2ADMNS and DB2USERS groups provide members with the following abilities:

- DB2ADMNS
Full control over all DB2 objects (see the list of protected objects, below)
- DB2USERS
Read and Execute access for all DB2 objects located in the installation and instance directories, but no access to objects under the database system directory and limited access to IPC resources

For certain objects, there may be additional privileges available, as required (for example, write privileges, add or update file privileges, and so on). Members of this group have no access to objects under the database system directory.

Note: The meaning of Execute access depends on the object; for example, for a **.dll** or **.exe** file having Execute access means you have authority to execute the file, however, for a directory it means you have authority to traverse the directory.

It is recommended that all DB2 administrators be members of the DB2ADMNS group (as well as being members of the local Administrators group), but this is not a strict requirement. Everyone else who requires access to the DB2 database system *must* be a member of the DB2USERS group. To add a user to one of these groups:

1. Launch the Users and Passwords Manager tool.
2. Select the user name to add from the list.
3. Click Properties. In the Properties window, click the Group membership tab.
4. Select the Other radio button.
5. Select the appropriate group from the drop-down list.

Adding extended security after installation (db2extsec command):

If the DB2 database system was installed without extended security enabled, you can enable it by executing the command **db2extsec** (called **db2secv82** in earlier releases). To execute the **db2extsec** command you must be a member of the local Administrators group so that you have the authority to modify the ACL of the protected objects.

You can run the **db2extsec** command multiple times, if necessary, however, if this is done, you cannot disable extended security unless you issue the **db2extsec -r** command immediately after *each* execution of **db2extsec**.

Removing extended security:

CAUTION:

It is not recommend to remove extended security once it has been enabled.

You can remove extended security by running the command **db2extsec -r**, however, this will only succeed if no other database operations (such as creating a database, creating a new instance, adding table spaces, and so on) have been performed after enabling extended security. The safest way to remove the extended security option is to uninstall the DB2 database system, delete all the relevant DB2 directories (including the database directories) and then reinstall the DB2 database system without extended security enabled.

Protected objects:

The *static* objects that can be protected using the DB2ADMNS and DB2USERS groups are:

- File system
 - File
 - Directory
- Services
- Registry keys

The *dynamic* objects that can be protected using the DB2ADMNS and DB2USERS groups are:

- IPC resources, including:
 - Pipes
 - Semaphores
 - Events
- Shared memory

Privileges owned by the DB2ADMNS and DB2USERS groups:

The privileges assigned to the DB2ADMNS and DB2USERS groups are listed in the following table:

Table 27. Privileges for DB2ADMNS and DB2USERS groups

Privilege	DB2ADMNS	DB2USERS	Reason
Create a token object (SeCreateTokenPrivilege)	Y	N	Token manipulation (required for certain token manipulation operations and used in authentication and authorization)
Replace a process level token (SeAssignPrimaryTokenPrivilege)	Y	N	Create process as another user
Increase quotas (SeIncreaseQuotaPrivilege)	Y	N	Create process as another user
Act as part of the operating system (SeTcbPrivilege)	Y	N	LogonUser (required prior to Windows XP in order to execute the LogonUser API for authentication purposes)
Generate security audits (SeSecurityPrivilege)	Y	N	Manipulate audit and security log
Take ownership of files or other objects (SeTakeOwnershipPrivilege)	Y	N	Modify object ACLs
Increase scheduling priority (SeIncreaseBasePriorityPrivilege)	Y	N	Modify the process working set
Backup files and directories (SeBackupPrivilege)	Y	N	Profile/Registry manipulation (required to perform certain user profile and registry manipulation routines: LoadUserProfile, RegSaveKey(Ex), RegRestoreKey, RegReplaceKey, RegLoadKey(Ex))
Restore files and directories (SeRestorePrivilege)	Y	N	Profile/Registry manipulation (required to perform certain user profile and registry manipulation routines: LoadUserProfile, RegSaveKey(Ex), RegRestoreKey, RegReplaceKey, RegLoadKey(Ex))
Debug programs (SeDebugPrivilege)	Y	N	Token manipulation (required for certain token manipulation operations and used in authentication and authorization)
Manage auditing and security log (SeAuditPrivilege)	Y	N	Generate auditing log entries
Log on as a service (SeServiceLogonRight)	Y	N	Run DB2 as a service

Table 27. Privileges for DB2ADMNS and DB2USERS groups (continued)

Privilege	DB2ADMNS	DB2USERS	Reason
Access this computer from the network (SeNetworkLogonRight)	Y	Y	Allow network credentials (allows the DB2 database manager to use the LOGON32_LOGON_NETWORK option to authenticate, which has performance implications)
Impersonate a client after authentication (SeImpersonatePrivilege)	Y	N	Client impersonation (required for Windows to allow use of certain APIs to impersonate DB2 clients: ImpersonateLoggedOnUser, ImpersonateSelf, RevertToSelf, and so on)
Lock pages in memory (SeLockMemoryPrivilege)	Y	N	AWE/Large Page support
Create global objects (SeCreateGlobalPrivilege)	Y	Y	Terminal Server support (required on Windows)

Related tasks:

- “Adding your user ID to the DB2ADMNS and DB2USERS user groups (Windows)” in *Quick Beginnings for DB2 Servers*

Related reference:

- “Required user accounts for installation of DB2 server products (Windows)” in *Quick Beginnings for DB2 Servers*
- “db2extsec - Set permissions for DB2 objects command” in *Command Reference*

UNIX platform security considerations for users

The DB2 database 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 platform security considerations for users” on page 485

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 /sql1ib sub-directory, in the directory where DB2 was installed.

Related concepts:

- “Instance creation” on page 34

Related tasks:

- “Creating additional instances” on page 38

Security plug-ins

Authentication in DB2® Universal Database (DB2 UDB) is done through *security plug-ins*. For more information, see Security plug-ins in the *Administrative API Reference*.

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.

Note: You can check the following web site for certification information on the cryptographic routines used by the DB2 database management system 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: http://www.ibm.com/security/standards/st_evaluations.shtml.

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 the DB2 database to set up a Common Criteria certified configuration, 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.

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.

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 user ID and password are provided and at the server when a user ID and password are provided.

Table 28. Authentication Modes using TRUST_ALLCLNTS and TRUST_CLNTAUTH Parameter Combinations.

TRUST_ALLCLNTS	TRUST_CLNTAUTH	Untrusted non-DRDA Client Authentication (no user ID & password)	Untrusted non-DRDA Client Authentication (with user ID & password)	Trusted non-DRDA Client Authentication (no user ID & password)	Trusted non-DRDA Client Authentication (with user ID & password)	DRDA Client Authentication (no user ID & password)	DRDA Client Authentication (with user ID & 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 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 database server. The KERBEROS authentication type is supported on clients and servers running Windows, 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 database 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 with DB2 UDB Version 7.1 and following.)
3. The client sends this service ticket to the server using 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 authentication 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, 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 and XQuery statements.
- SQL program variable data.
- Output data from the server processing of an SQL or XQuery 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 through 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-supplied Kerberos plug-in exists for the operating system, or a Kerberos-based plug-in is specified for the *srvcn_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
- SYSMANT_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 database system, you have a fail-safe option available on all platforms that will allow you to override the usual DB2 database 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 database command. This special user is identified as follows:

- UNIX platforms: the instance owner
- Windows 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 495
- “DB2 and Windows security introduction” on page 675
- “Partitioned database authentication considerations” on page 496

Related reference:

- “authentication - Authentication type configuration parameter” in *Performance Guide*
- “trust_allclnts - Trust all clients configuration parameter” in *Performance Guide*
- “trust_clntauth - Trusted clients authentication configuration parameter” in *Performance Guide*

Authentication considerations for remote clients

When cataloging a database for remote access, the authentication type can be specified in the database directory entry.

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 database attempts to recover. Recovery may result in more flows to reconcile the difference, or in an error if the DB2 database cannot recover. In the case of a mismatch, the value at the server is assumed to be correct.

The authentication type DATA_ENCRYPT_CMP is designed to allow clients from a previous release that does not support data encryption to a server using SERVER_ENCRYPT authentication instead of DATA_ENCRYPT. This authentication does not work when the following statements are true:

- The client level is Version 7.2.
- The gateway level is Version 8 FixPak7 or later.
- The server is Version 8 FixPak 7 or later.

When these are all true, the client cannot connect to the server. To allow the connection, you must either upgrade your client to Version 8, or have your gateway level at Version 8 FixPak 6 or earlier.

The determination of the authentication type used when connecting is made by specifying the appropriate authentication type as a database catalog entry at the gateway. This is true for both DB2 Connect scenarios and for clients and servers in a partitioned database environment where the client has set the DB2NODE registry variable. You will catalog the authentication type at the catalog partition with the intent to “hop” to the appropriate partition. In this scenario, the authentication type cataloged at the gateway is not used because the negotiation is solely between the client and the server.

You may have a need to catalog multiple database aliases at the gateway using different authentication types if they need to have clients that use differing authentication types. When deciding which authentication type to catalog at a gateway, you can keep the authentication type the same as that used at the client and server; or, you can use the NOTSPEC authentication type with the understanding that NOTSPEC defaults to SERVER.

Related concepts:

- “Authentication methods for your server” on page 490

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 490

Kerberos authentication details

The DB2 database system provides support for the Kerberos authentication protocol on AIX, Solaris, Linux IA32 and AMD64, and Windows 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 database system.

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. 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 database, this means that a user is able to connect or attach to a DB2 database 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 database system 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 database. 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 database system 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 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 database 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 database 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 database 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 database 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 database 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 database instance since the server name is reported to DB2 database by the plugin at initialization time.

On Windows, the server principal is taken to be the domain account under which the DB2 database 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.)

You can override the Kerberos server principal name used by the DB2 server on UNIX and Linux operating systems. Set the `DB2_KRB5_PRINCIPAL` environment

variable to the desired fully qualified server principal name. The instance must be restarted because the server principal name is only recognized by the DB2 database system after **db2start** is run.

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 database 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 database 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 database 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 database 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 *svrcon_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 database 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 database engine process, this environment variable may not be accessible.

Linux prerequisites:

The provided DB2 Kerberos security plug-in is supported with Red Hat Enterprise Linux Advanced Server 3 with the IBM Network Authentication Service (NAS) 1.4 client.

zSeries and iSeries compatibility:

For connections to zSeries and iSeries, the database must be cataloged with the AUTHENTICATION KERBEROS parameter and the TARGET PRINCIPAL parameter name must be explicitly specified.

Neither zSeries nor iSeries support mutual authentication.

Windows issues:

When you are using Kerberos on Windows platforms, you need to be aware of the following issues:

- Due to the manner in which Windows detects and reports some errors, the following conditions result in an unexpected client security plug-in error (SQL30082N, rc=36):
 - Expired account
 - Invalid password
 - Expired password
 - Password change forced by administrator
 - Disabled account

Furthermore, in all cases, the DB2 administration log or db2diag.log will indicate "Logon failed" or "Logon denied".

- If a domain account name is also defined locally, connections explicitly specifying the domain name and password will fail with the following error: The Local Security Authority cannot be contacted.

The error is a result of Windows locating the local user first. The solution is to fully qualify the user in the connection string. For example:
name@DOMAIN.IBM.COM

- Windows accounts cannot include the @ character in their name because the character is assumed to be the domain separator by the DB2 Kerberos plug-in.
- When interoperating with a non-Windows platform, ensure that all Windows domain server accounts and all Windows client accounts are configured to use DES encryption. If the account used to start the DB2 service is not configured to use DES encryption, the DB2 server will fail to accept Kerberos contexts. In particular, DB2 will fail with an unexpected server plug-in error, and will log that the AcceptSecurityContext API returned SEC_I_CONTINUE_NEEDED (0x00090312L).

To determine if Windows accounts are configured to use DES encryption, look under **Account properties** in the **Active Directory**. A restart might be required if the account properties are changed.

- If the client and server are both on Windows, then the DB2 service can be started under the local system account. However, if the client and server are in different domains, the connection might fail with an invalid target principal name error. The workaround is to explicitly catalog the target principal name on the client using the fully qualified server host name and the fully qualified domain name, in the following format: `host/server hostname@server domain name`
For example: `host/myhost.domain.ibm.com@DOMAIN.IBM.COM`

Otherwise, you must start the DB2 service under a valid domain account.

Related concepts:

- “Authentication methods for your server” on page 490

Authorization, privileges, and object ownership

Users (identified by an authorization ID) can successfully execute SQL or XQuery statements only if they have the authority to perform the specified function. 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 forth.

There are two forms of authorization, *administrative authority* and *privileges*, discussed below.

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).

Administrative authority:

The person or persons holding administrative authority are charged with the task of controlling the database manager and are responsible for the safety and integrity of the data. Those with administrative authority levels of SYSADM and DBADM implicitly have all privileges on all objects except objects pertaining to database security and control who will have access to the database manager and the extent of this access.

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. A user or group can have one or more of the following authorities:

- Administrative authority level that operates at the instance level, 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, SYSMANT, and SYSMON, and the authority to grant and revoke DBADM authority and SECADM 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 DBADM authority within a database but does not provide implicit SECADM authority within a database.

- Administrative authority levels that operate at the database level:

- DBADM (database administrator)

The DBADM authority level applies at the database level and 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.

- SECADM (security administrator)

The SECADM authority level applies at the database level and is the authority required to create and drop security label components, security policies, and security labels, which are used to protect tables. It is also the authority required to grant and revoke security labels and exemptions as well as to grant and revoke the SETSESSIONUSER privilege. A user with the SECADM authority can transfer the ownership of objects that they do not own. The SECADM authority has no inherent privilege to access data stored in tables and has no other additional inherent privilege. It can only be granted by a user with SYSADM authority. The SECADM authority can be granted to a user but cannot be granted to a group or to PUBLIC.

- System control authority levels that operate at the instance level:

- 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, start, stop, or drop a database. This user can also start or stop an instance, but cannot access table data. Users with SYSCTRL authority also have SYSMON authority.

- SYSMANT (system maintenance)

The SYSMANT authority level provides the authority required to perform maintenance operations on all databases associated with an instance. A user with SYSMANT authority can update the database configuration, backup a database or table space, restore an existing database, and monitor a database. Like SYSCTRL, SYSMANT does not provide access to table data. Users with SYSMANT authority also have SYSMON authority.

- The SYSMON (system monitor) authority level

SYSMON provides the authority required to use the database system monitor. It operates at the instance level.

- 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 example, the LOAD

database authority is required for use of the load utility to load data into tables (a user must also have INSERT privilege on the table).

Figure 5 illustrates the relationship between authorities and their span of control (database, database manager).

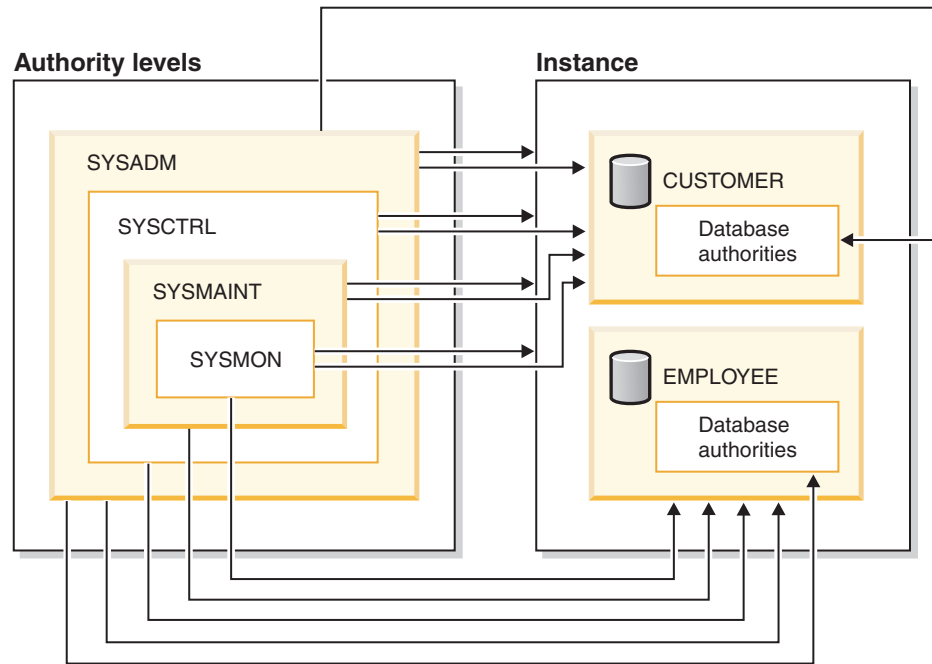


Figure 5. Hierarchy of Authorities

Privileges:

Privileges are those activities that a user is allowed to perform. Authorized users can create objects, have access to objects they own, and can pass on privileges on their own objects to other users by using the GRANT statement.

Privileges may be granted to individual users, to groups, or to PUBLIC. PUBLIC is a special group that consists of all users, including future users. Users that are members of a group will indirectly take advantage of the privileges granted to the group, where groups are supported.

The 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 applies 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 could grant the CONTROL privilege to that object. The CONTROL privilege cannot be revoked from the object owner, however, the object owner can be changed by using the TRANSFER OWNERSHIP statement.

In some situations, the creator of an object automatically obtains the CONTROL privilege on that object.

Individual 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 on objects in a package or routine: When a user has the privilege to execute a package or routine, they do not necessarily require specific privileges on the objects used in the package or routine. If the package or routine contains static SQL or XQuery statements, the privileges of the owner of the package are used for those statements. If the package or routine contains dynamic SQL or XQuery statements, the authorization ID used for privilege checking depends on the setting of the DYNAMICRULES bind option of the package issuing the dynamic query statements, and whether those statements are issued when the package is being used in the context of a routine.

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 representing a user or group is granted authorities and privileges and there is no user or group created with that name. At some later time, a user or group can be created with that 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. 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.

Object ownership:

When an object is created, one authorization ID is assigned *ownership* of the object. Ownership means the user is authorized to reference the object in any applicable SQL or XQuery 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 (C1 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.

Certain privileges on the object, such as altering a table, can be granted by the owner, and can be revoked from the owner by a user who has SYSADM or DBADM authority. Certain privileges on the object, such as commenting on a table, cannot be granted by the owner and cannot be revoked from the owner. Use the TRANSFER OWNERSHIP statement to move these privileges to another user. When an object is created, the authorization ID of the statement is the owner of the object. However, when a package is created and the OWNER bind option is specified, the owner of objects created by the static SQL statements in the package is the value of the OWNER bind option. In addition, if the AUTHORIZATION clause is specified on a CREATE SCHEMA statement, the authorization name specified after the AUTHORIZATION keyword is the owner of the schema.

A security administrator (SECADM) or the object owner can use the TRANSFER OWNERSHIP statement to change the ownership of a database object. An administrator can therefore create an object on behalf of an authorization ID, by creating the object using the authorization ID as the qualifier, and then using the

TRANSFER OWNERSHIP statement to transfer the ownership that the administrator has on the object to the authorization ID.

Related concepts:

- “Controlling access to database objects” on page 519
- “Database administration authority (DBADM)” on page 509
- “Database authorities” on page 511
- “Index privileges” on page 518
- “Indirect privileges through a package” on page 523
- “LOAD authority” on page 511
- “Package privileges” on page 517
- “Routine privileges” on page 518
- “Schema privileges” on page 514
- “Security administration authority (SECADM)” on page 508
- “Sequence privileges” on page 518
- “System administration authority (SYSADM)” on page 506
- “System control authority (SYSCTRL)” on page 507
- “System maintenance authority (SYSMAINT)” on page 508
- “System monitor authority (SYSMON)” on page 510
- “Table and view privileges” on page 515
- “Table space privileges” on page 515

Related reference:

- “GRANT (Database Authorities) statement” in *SQL Reference, Volume 2*

Details on privileges, authorities, and authorization

This section discusses each of the authorities and 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 any data that is not protected by LBAC 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 and revoke DBADM authority.
- Grant and revoke SECADM authority

While SYSADM authority does provide all abilities provided by most other authorities, it does not provide any of the abilities of the SECADM authority. The abilities provided by the SECADM authority are not provided by any other authority. SYSADM authority also does not provide access to data that is protected by LBAC.

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:

- "Data encryption" on page 527
- "Security administration authority (SECADM)" on page 508
- "System control authority (SYSCTRL)" on page 507
- "System maintenance authority (SYSMAINT)" on page 508
- "System monitor authority (SYSMON)" on page 510

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:

- “Database administration authority (DBADM)” on page 509
- “System maintenance authority (SYSMAINT)” on page 508
- “System monitor authority (SYSMON)” on page 510

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 509
- “System monitor authority (SYSMON)” on page 510

Security administration authority (SECADM)

SECADM authority can only be granted by the SYSADM and can be granted to a user but not to a group. It gives these and only these abilities:

- Create and drop security label components

- Create and drop security policies
- Create and drop security labels
- Grant and revoke security labels
- Grant and revoke LBAC rule exemptions
- Grant and revoke setsessionuser privileges
- Execute the SQL statement TRANSFER OWNERSHIP on objects that you do not own

No other authority gives these abilities, not even SYSADM.

Related concepts:

- “Database authorities” on page 511
- “System administration authority (SYSADM)” on page 506
- “System control authority (SYSCTRL)” on page 507
- “System maintenance authority (SYSMAINT)” on page 508

Database administration authority (DBADM)

DBADM authority is an administrative authority for a specific database and it allows the user to perform certain actions, and issue database commands on that database. The DBADM authority allows access to the data in any table in the database unless that data is protected by LBAC. To access data protected by LBAC you must have appropriate LBAC credentials.

When DBADM authority is granted, the following database authorities are also explicitly granted for the same database (and are not automatically revoked if the DBADM authority is later revoked):

- BINDADD
- CONNECT
- CREATETAB
- CREATE_EXTERNAL
- ROUTINE, CREATE
- NOT_FENCED_ROUTINE
- IMPLICIT_SCHEMA
- QUIESCE_CONNECT
- LOAD

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.

Holding the DBADM, or higher, authority for a database allows a user to perform these actions on that database:

- Read log files
- Create, activate, and drop event monitors.

A user with DBADM authority for a database or with SYSMAINT authority or higher can perform these actions on the database:

- 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.

While DBADM authority does provide some of the same abilities as other authorities, it does not provide any of the abilities of the SECADM authority. The abilities provided by the SECADM authority are not provided by any other authority.

Related concepts:

- “Database authorities” on page 511
- “Implicit schema authority (IMPLICIT_SCHEMA) considerations” on page 513
- “LOAD authority” on page 511
- “Security administration authority (SECADM)” on page 508
- “System administration authority (SYSADM)” on page 506
- “System control authority (SYSCTRL)” on page 507
- “System maintenance authority (SYSMAINT)” on page 508

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.SNAP_WRITE_FILE`
`SYSPROC.SNAP_WRITE_FILE` 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 *Performance Guide*

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:

- “Table and view privileges” on page 515
- “Privileges, authorities, and authorizations required to use Load” in *Data Movement Utilities Guide and Reference*

Related reference:

- “LIST TABLESPACES command” in *Command Reference*
- “LOAD command” in *Command Reference*
- “QUIESCE TABLESPACES FOR TABLE command” in *Command Reference*
- “RUNSTATS command” in *Command Reference*

Database authorities

Each database authority allows the authorization ID holding it to perform some particular type of action on the database as a whole. Database authorities are different from privileges, which allow a certain action to be taken on a particular database object, such as a table or an index. There are ten different database authorities.

SECADM

Gives the holder the ability to configure many things related to security of the database, and also to transfer ownership of database objects. For instance, all objects that are part of the label-based access control (LBAC) feature can be created, dropped, granted, or revoked by a user that holds SECADM authority. SECADM specific abilities cannot be exercised by any other authority, not even SYSADM.

DBADM

Gives the holder the authority to act as the database administrator. In particular it gives the holder all of the other database authorities except for SECADM.

CONNECT

Allows the holder to connect to the database.

BINDADD

Allows the holder to create new packages in the database.

CREATETAB

Allows the holder to create new tables in the database.

CREATE_EXTERNAL_ROUTINE

Allows the holder to create a procedure for use by applications and other users of the database.

CREATE_NOT_FENCED_ROUTINE

Allows the holder to create a user-defined function (UDF) or procedure that is “not fenced”. CREATE_EXTERNAL_ROUTINE is automatically granted to any user who is granted CREATE_NOT_FENCED_ROUTINE.

Attention:: The database manager does not protect its storage or control blocks from UDFs or procedures that are “not fenced”. A user with this authority must, therefore, be very careful to test their UDF extremely well before registering it as “not fenced”.

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 the holder to load data into a table

QUIESCE_CONNECT

Allows the holder to access the database while it is quiesced.

Only authorization IDs with the SYSADM authority can grant the SECADM and DBADM authorities. All other authorities can be granted by authorization IDs that hold SYSADM or DBADM authorities.

When a database is created, the following database authorities are automatically granted to PUBLIC for the new database:

- CREATETAB
- BINDADD
- CONNECT
- IMPLICIT_SCHEMA

In addition, these privileges are granted:

- USE privilege on USERSPACE1 table space
- SELECT privilege on the system catalog views.

To remove any database authority from PUBLIC, an authorization ID with DBADM or SYSADM authority must explicitly revoke it.

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

Authorization ID privileges

Authorization ID privileges involve actions on authorization IDs. There is currently only one such privilege: the SETSESSIONUSER privilege.

The SETSESSIONUSER privilege can be granted to a user or to a group and allows the holder to switch identities to any of the authorization IDs on which the privilege was granted. The identity switch is made by using the SQL statement SET SESSION AUTHORIZATION. The SETSESSIONUSER privilege can only be granted by a user holding SECADM authority.

Note: When you migrate a DB2 UDB database to DB2 Version 9.1, authorization IDs with explicit DBADM authority on that database will automatically be granted SETSESSIONUSER privilege on PUBLIC. This prevents breaking applications that rely on authorization IDs with DBADM authority being able to set the session authorization ID to any authorization ID. This does not happen when the authorization ID has SYSADM authority but has not been explicitly granted DBADM.

Related concepts:

- “Authorization, privileges, and object ownership” on page 501

Related reference:

- “SET SESSION AUTHORIZATION statement” in *SQL Reference, Volume 2*

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 519
- “Revoking privileges” on page 521

Schema privileges

Schema privileges are in the object privilege category. Object privileges are shown in Figure 6.

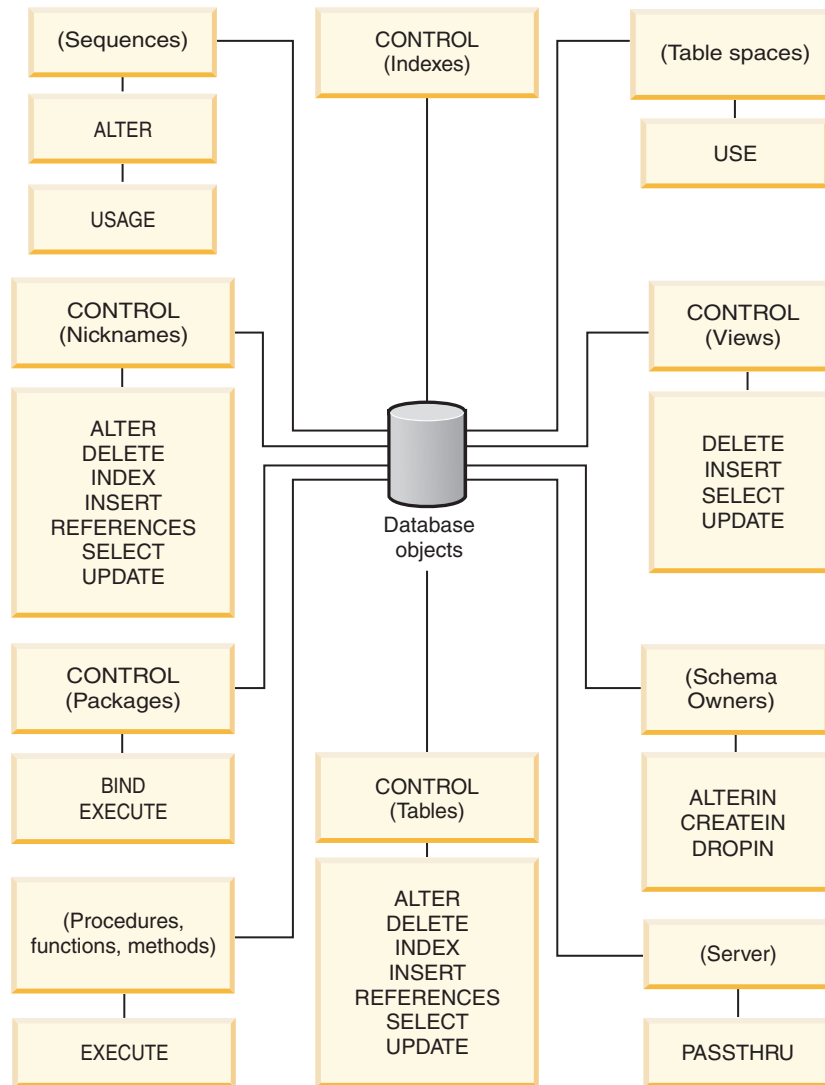


Figure 6. Object Privileges

Schema privileges involve actions on schemas in a database. A user may be granted any of the following privileges:

- CREATEIN allows the user to create objects within the schema.
- ALTERIN allows the user to alter objects within the schema.
- DROPIN allows the user to drop objects from within the schema.

The owner of the schema has all of these privileges and the ability to grant them to others. The objects that are manipulated within the schema object include: tables, views, indexes, packages, data types, functions, triggers, procedures, and aliases.

Related tasks:

- “Granting privileges” on page 519

- “Revoking privileges” on page 521

Related reference:

- “ALTER SEQUENCE statement” in *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 519
- “Revoking privileges” on page 521

Related reference:

- “CREATE TABLE statement” in *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 518

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE VIEW statement” in *SQL Reference, Volume 2*
- “SELECT statement” in *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 database uses dynamic queries 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 511

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

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 515

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

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 519
- “Revoking privileges” on page 521

Related reference:

- “ALTER SEQUENCE statement” in *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 519
- “Revoking privileges” on page 521

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 609

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

Details on controlling access to database objects

This section describes the control of access to database objects through the use of GRANT and REVOKE statements. It also discusses implicit access and indirect privileges.

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
```

In the Control Center, you can use the Schema Privileges notebook, the Table Space Privileges notebook, and the View Privileges notebook to grant and revoke privileges for these database objects. To open one of these notebooks, follow these steps:

1. In the Control Center, expand the object tree until you find the folder containing the objects you want to work with, for example, the Views folder.
2. Click the folder.
Any existing database objects in this folder are displayed in the contents pane.
3. Right-click the object of interest in the contents pane and select **Privileges** in the pop-up menu.
The appropriate Privileges notebook opens.

Related concepts:

- “Controlling access to database objects” on page 519

Related tasks:

- “Revoking privileges” on page 521

Related reference:

- “GRANT (Database Authorities) statement” in *SQL Reference, Volume 2*
- “GRANT (Index Privileges) statement” in *SQL Reference, Volume 2*
- “GRANT (Package Privileges) statement” in *SQL Reference, Volume 2*
- “GRANT (Routine Privileges) statement” in *SQL Reference, Volume 2*
- “GRANT (Schema Privileges) statement” in *SQL Reference, Volume 2*
- “GRANT (Sequence Privileges) statement” in *SQL Reference, Volume 2*
- “GRANT (Server Privileges) statement” in *SQL Reference, Volume 2*
- “GRANT (Table Space Privileges) statement” in *SQL Reference, Volume 2*
- “GRANT (Table, View, or Nickname Privileges) statement” in *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 519

Related reference:

- “REVOKE (Database Authorities) statement” in *SQL Reference, Volume 2*
- “REVOKE (Index Privileges) statement” in *SQL Reference, Volume 2*
- “REVOKE (Package Privileges) statement” in *SQL Reference, Volume 2*
- “REVOKE (Routine Privileges) statement” in *SQL Reference, Volume 2*
- “REVOKE (Schema Privileges) statement” in *SQL Reference, Volume 2*
- “REVOKE (Server Privileges) statement” in *SQL Reference, Volume 2*
- “REVOKE (Table Space Privileges) statement” in *SQL Reference, Volume 2*
- “REVOKE (Table, View, or Nickname Privileges) statement” in *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 519
- “Revoking privileges” on page 521

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 database products support the OWNER option.

Related reference:

- “BIND command” in *Command Reference*
- “PRECOMPILE command” in *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 and XQuery statements are bound. Privileges granted through groups are *not* used for authorization checking when static SQL and XQuery statements are 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 or XQuery 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 or XQuery statements within this package will not cause the BIND to fail, and those SQL or XQuery 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 and XQuery statements. To process a package with static queries, 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 queries in the package but only within the restrictions imposed by the package.

If the package includes dynamic SQL or XQuery statements, 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 queries.

Related concepts:

- “Indirect privileges through a package containing nicknames” on page 524
- “Effect of DYNAMICRULES bind option on dynamic SQL” in *Developing Embedded SQL Applications*

Related reference:

- “BIND command” in *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 .SQL file contains several SQL or XQuery 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 .SQL file contains only dynamic SQL and XQuery statements and a mixture of table and nickname references, DB2 database 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 and XQuery statements. 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 database 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 523

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 database 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

NAME	LOCATION
O'Brien	Atlanta
Quigley	Atlanta
Naughton	Atlanta
Abrahams	Atlanta
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver
Davis	Denver
Edwards	Denver
Gafney	Denver

Related tasks:

- “Creating a view” on page 251
- “Granting privileges” on page 519

Monitoring access to data using the audit facility

The DB2 database 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 database audit facility” on page 621

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:

- "DECRYPT_BIN and DECRYPT_CHAR scalar functions" in *SQL Reference, Volume 1*
- "ENCRYPT scalar function" in *SQL Reference, Volume 1*
- "GETHINT scalar function" in *SQL Reference, Volume 1*
- "SET ENCRYPTION PASSWORD statement" in *SQL Reference, Volume 2*

Granting database authorities to new groups

You can use the Database page of the Add Group notebook to grant authorities on a database to a new group of users of this database.

Prerequisites:

To grant database authorities, you need the proper authorizations:

- To grant BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA authorities, you need either SYSADM or DBADM authority.
- To grant DBADM authority, you need SYSADM authority.

Procedure:

To grant database authorities:

1. Under **Choose the appropriate authorities to grant to the selected group**, indicate which database authority or authorities you want to grant.
2. If you didn't specify the authorization name of the group to which you're granting database authorities, use the **Group** box to do so.
3. Click **Apply**. The authority or authorities are granted.
4. Optional: Grant one or more of the following types of privileges, in any order:
 - Schema privileges
 - Table privileges
 - Index privileges
 - View privileges
 - Table Space privileges
 - Function privileges
 - Procedure privileges
 - Package privileges
 - Method privileges

Related tasks:

- "Granting database authorities to new users" on page 529
- "Granting privileges to new groups" on page 530
- "Granting privileges to new users" on page 534

Granting database authorities to new users

You can use the Database page of the Add User notebook to grant authorities on a database to a new user of the database.

Prerequisites:

To grant database authorities, you need the proper authorizations:

- To grant BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA authorities, you need either SYSADM or DBADM authority.
- To grant DBADM authority, you need SYSADM authority.

Procedure:

To grant database authorities:

1. Open the Add User notebook.
2. Under **Choose the appropriate authorities to grant to the selected user**, indicate which database authority or authorities you want to grant.
3. If you didn't specify the authorization name of the user to whom you are granting database authorities, you must use the **User** box to do so.
4. Click **Apply**. The authority or authorities are granted.
5. Optional: Grant one or more of the following types of privileges, in any order:
 - Schema privileges
 - Table privileges
 - Index privileges
 - View privileges
 - Table Space privileges
 - Function privileges
 - Procedure privileges
 - Method privileges
 - Package privileges

Related tasks:

- "Granting privileges to new users" on page 534
- "Granting database authorities to new groups" on page 529
- "Granting privileges to new groups" on page 530

Granting privileges to new groups

Use the Add Group notebook to authorize a group of users to use a database and the objects in it.

Prerequisites:

Task	Authorities and privileges
To grant or revoke database authorities	You need the proper authorizations: <ul style="list-style-type: none"> • To grant BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA authorities, you need either SYSADM or DBADM authority. • To grant DBADM authority, you need SYSADM authority.

Task	Authorities and privileges
To grant or revoke a privilege on a schema	<p>You need one of the following authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority • The privilege with the Grant option (the right to grant the privilege to other groups and users) <p>Example</p> <p>You can grant the ALTERIN privilege on a schema if you have one of these authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority on the database in which the schema resides • The ALTERIN privilege on the schema, along with the right to grant the ALTERIN privilege on the schema to others
To grant or revoke privileges on tables or views	<p>You need the proper authorizations:</p> <ul style="list-style-type: none"> • To grant or revoke privileges on catalog tables and views, you need either SYSADM or DBADM authority. • To grant or revoke privileges on user-defined tables and views, you need to meet the following requirements: <ul style="list-style-type: none"> - To grant the CONTROL privilege on a table or view, you need SYSADM or DBADM authority. - To grant table or view privileges other than CONTROL, you need one of the following authorizations. To revoke table or view privileges other than CONTROL, you need one of the first three of these authorizations: <ul style="list-style-type: none"> - SYSADM authority - DBADM authority - The CONTROL privilege on the tables or views that you want to grant privileges on - The privilege that you want to grant, with the Grant option (the right to grant the privilege to other groups and users) <p>Example</p> <p>You can grant the ALTER privilege on a user-defined table if you hold one of these authorizations:</p> <ul style="list-style-type: none"> - SYSADM authority - DBADM authority on the database in which the table resides - The CONTROL privilege on the table - The ALTER privilege, along with the right to grant the ALTER privilege on this table to other groups and users
To grant or revoke the CONTROL privilege on an index	You need either SYSADM authority or DBADM authority.

Task	Authorities and privileges
To authorize a group to use a database	<p>You need one of the following authorizations:</p> <ul style="list-style-type: none"> • Authorization to grant database authorities • Authorization to grant schema privileges • Authorization to grant table or view privileges • Authorization to grant the CONTROL privilege on indexes • Authorization to grant package privileges • Authorization to grant routine (function, methods, and procedures) privileges
To grant database authorities	<p>You need the proper authorizations:</p> <ul style="list-style-type: none"> • To grant BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA authorities, you need either SYSADM or DBADM authority. • To grant DBADM authority, you need SYSADM authority.
To grant a privilege on a schema	<p>You need one of the following authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority • The privilege with the Grant option (that is, with the right to grant the privilege to other groups and users) <p>Example:</p> <p>You can grant the ALTERIN privilege on a schema if you have one of these authorities:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority on the database in which the schema resides • The ALTERIN privilege on the schema, along with the right to grant the ALTERIN privilege on the schema to others

Task	Authorities and privileges
To grant privileges on tables or views	<p>You need the proper authorizations:</p> <ul style="list-style-type: none"> • To grant privileges on catalog tables and views, you need either SYSADM or DBADM authority. • To grant privileges on user-defined tables and views, you need to meet the following requirements: <ul style="list-style-type: none"> – To grant the CONTROL privilege on a table or view, you need SYSADM or DBADM authority. – To grant table or view privileges other than CONTROL, you need one of these authorities: <ul style="list-style-type: none"> - SYSADM authority - DBADM authority - The CONTROL privilege on the tables or views that you want to grant privileges on - The privilege you want to grant, along with the Grant option (the right to grant this privilege to other groups and users) <p>Example</p> <p>You can grant the ALTER privilege on a user-defined table if you hold one of these authorities:</p> <ul style="list-style-type: none"> - SYSADM authority - DBADM authority on the database in which the table resides - The CONTROL privilege on the table - The ALTER privilege, along with the right to grant the ALTER privilege on this table to other people
To grant the CONTROL privilege on an index	You need either SYSADM authority or DBADM authority.
To grant or revoke privileges on a package	<p>To grant or revoke the BIND and EXECUTE privilege, you must have one of the following authorizations:</p> <ul style="list-style-type: none"> • CONTROL privilege on the referenced package • SYSADM authority • DBADM authority on the database <p>To grant or revoke the CONTROL privilege, SYSADM or DBADM authority is required.</p>
To grant or revoke privileges on a routine (function, methods, and procedures)	<p>You must have the following authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority, DBADM authority, or a user with the GRANT option on EXECUTE on a routine, can grant the EXECUTE privilege on that routine. SYSADM or DBADM authority can revoke the EXECUTE privilege on a routine. • The EXECUTE privilege cannot be granted or revoked on functions in the SYSIBM and SYSFUN schemas. Functions in these schemas are considered to have the equivalent of EXECUTE WITH GRANT OPTION granted to PUBLIC, allowing public use of these functions in SQL routines and sourced functions.

Procedure:

1. Open the Add Group notebook: From the Control Center, expand the object tree until you find the **Databases** folder. Open the **Databases** folder. Any existing databases are displayed in the object tree. Click the database you want and locate the **User and Group Objects** folder. Click the **User and Group**

- Objects** folder. The DB Groups folder appears. Right-click the **DB Groups** folder and select **Add** from the pop-up menu. The Add Group notebook opens.
2. On the Database page, define a new group of database users to DB2. Use the **Group** box to specify the group's authorization name.
 3. Grant this group one or more of the following types of authorizations, in any order:
 - Database authorities
 - Schema privileges
 - Table privileges
 - Index privileges
 - View privileges
 - Table Space privileges
 - Function privileges
 - Procedure privileges
 - Method privileges
 - Package privileges

For more information, refer to the online help.

Related concepts:

- “Authorization ID privileges” on page 513
- “Authorization, privileges, and object ownership” on page 501
- “Controlling access to database objects” on page 519

Related tasks:

- “Revoking privileges” on page 521
- “Granting database authorities to new groups” on page 529
- “Granting privileges” on page 519

Granting privileges to new users

Use the Add User notebook to authorize a user to use a database and the objects in it.

Prerequisites:

Task	Authorities and privileges
To grant or revoke database authorities	You need the proper authorizations: <ul style="list-style-type: none"> • To grant BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA authorities, you need either SYSADM or DBADM authority. • To grant DBADM authority, you need SYSADM authority.

Task	Authorities and privileges
To grant or revoke a privilege on a schema	<p>You need one of the following authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority • The privilege with the Grant option (the right to grant the privilege to other users and to groups) <p>Example</p> <p>You can grant the ALTERIN privilege on a schema if you have one of these authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority on the database in which the schema resides • The ALTERIN privilege on the schema, along with the right to grant the ALTERIN privilege on the schema to other users and to groups
To grant or revoke privileges on tables or views	<p>You need the proper authorizations:</p> <ul style="list-style-type: none"> • To grant or revoke privileges on catalog tables and views, you need either SYSADM or DBADM authority. • To grant or revoke privileges on user-defined tables and views, you need to meet the following requirements: <ul style="list-style-type: none"> – To grant or revoke the CONTROL privilege on a table or view, you need SYSADM or DBADM authority. – To grant table or view privileges other than CONTROL, you need one of the following authorizations. To revoke table or view privileges other than CONTROL, you need one of the first three of these authorizations: <ul style="list-style-type: none"> - SYSADM authority - DBADM authority - The CONTROL privilege on the tables or views that you want to grant privileges on - The privilege you want to grant with the Grant option (the right to grant the privilege to other users and to groups) <p>Example</p> <p>You can grant the ALTER privilege on a user-defined table if you hold one of these authorizations:</p> <ul style="list-style-type: none"> - SYSADM authority - DBADM authority on the database in which the table resides - The CONTROL privilege on the table - The ALTER privilege, along with the right to grant the ALTER privilege on this table to other users and to groups
To grant or revoke the CONTROL privilege on an index	You need either SYSADM authority or DBADM authority.

Task	Authorities and privileges
To define a person to DB2 as a user of a database	<p>You need one of the following authorizations:</p> <ul style="list-style-type: none"> • Authorization to grant database authorities • Authorization to grant schema privileges • Authorization to grant table or view privileges • Authorization to grant the CONTROL privilege on indexes • Authorization to grant package privileges • Authorization to grant routine (function, methods, and procedures) privileges
To grant database authorities	<p>You need the proper authorizations:</p> <ul style="list-style-type: none"> • To grant BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA authorities, you need either SYSADM or DBADM authority. • To grant DBADM authority, you need SYSADM authority.
To grant a privilege on a schema	<p>You need one of the following authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority • The privilege with the Grant option (that is, with the right to grant the privilege to other users and to groups) <p>Example</p> <p>You can grant the ALTERIN privilege on a schema if you have one of these authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority • DBADM authority on the database in which the schema resides • The ALTERIN privilege on the schema, along with the right to grant the ALTERIN privilege on the schema to other users and to groups

Task	Authorities and privileges
To grant privileges on tables or views	<p>You need the proper authorizations:</p> <ul style="list-style-type: none"> • To grant privileges on catalog tables and views, you need either SYSADM or DBADM authority. • To grant privileges on user-defined tables and views, you need to meet the following requirements: <ul style="list-style-type: none"> – To grant the CONTROL privilege on a table or view, you need SYSADM or DBADM authority. – To grant table or view privileges other than CONTROL, you need one of these authorizations: <ul style="list-style-type: none"> - SYSADM authority - DBADM authority - The CONTROL privilege on the tables or views that you want to grant privileges on - The privilege you want to grant, along with the Grant option (the right to grant this privilege to other users and to groups) <p>Example</p> <p>You can grant the ALTER privilege on a user-defined table if you hold one of these authorities:</p> <ul style="list-style-type: none"> - SYSADM authority - DBADM authority on the database in which the table resides - The CONTROL privilege on the table - The ALTER privilege, along with the right to grant the ALTER privilege on this table to other users and to groups
To grant the CONTROL privilege on an index	You need either SYSADM authority or DBADM authority.
To grant or revoke privileges on a package	<p>To grant or revoke the BIND and EXECUTE privilege, you must have one of the following authorizations:</p> <ul style="list-style-type: none"> • CONTROL privilege on the referenced package • SYSADM authority • DBADM authority on the database <p>To grant or revoke the CONTROL privilege, SYSADM or DBADM authority is required.</p>
To grant or revoke privileges on a routine (function, methods, and procedures)	<p>You must have the following authorizations:</p> <ul style="list-style-type: none"> • SYSADM authority, DBADM authority, or a user with the GRANT option on EXECUTE on a routine, can grant the EXECUTE privilege on that routine. SYSADM or DBADM authority can revoke the EXECUTE privilege on a routine. • The EXECUTE privilege cannot be granted or revoked on functions in the SYSIBM and SYSFUN schemas. Functions in these schemas are considered to have the equivalent of EXECUTE WITH GRANT OPTION granted to PUBLIC, allowing public use of these functions in SQL routines and sourced functions.

Procedure:

1. Open the Add User notebook: From the Control Center window, expand the object tree until you find the **User and Group Objects** folder below the database that you're authorizing a user to use. Click on this folder. The **DB**

Users folder appears. Right-click the **DB Users** folder and select **Add** from the pop-up menu. The Add User notebook opens.

2. Use the **User** box to specify the user's authorization name.
3. Grant this user one or more of the following types of authorizations, in any order:
 - Database authorities
 - Schema privileges
 - Table privileges
 - Index privileges
 - View privileges
 - Table Space privileges
 - Function privileges
 - Procedure privileges
 - Method privileges
 - Package privileges

For more information, refer to the online help.

Related concepts:

- "Authorization ID privileges" on page 513
- "Authorization, privileges, and object ownership" on page 501
- "Controlling access to database objects" on page 519

Related tasks:

- "Granting database authorities to new users" on page 529
- "Granting privileges" on page 519
- "Retrieving all privileges granted to users" on page 613
- "Revoking privileges" on page 521

Label-based access control (LBAC)

This section explains what you need to know in order to use label-based access control (LBAC).

Label-based access control (LBAC) overview

What LBAC does:

Label-based access control (LBAC) greatly increases the control you have over who can access your data. LBAC lets you decide exactly who has write access and who has read access to individual rows and individual columns.

The LBAC capability is very configurable and can be tailored to match your particular security environment. All LBAC configuration is performed by a *security administrator*, which is a user that has been granted the SECADM authority by the system administrator.

A security administrator configures the LBAC system by creating security policies. A *security policy* describes the criteria that will be used to decide who has access to what data. Only one security policy can be used to protect any one table but different tables can be protected by different security policies.

After creating a security policy, a security administrator creates objects, called *security labels* that are part of that policy. Exactly what makes up a security label is determined by the security policy and can be configured to represent the criteria that your organization uses to decide who should have access to particular data items. If you decide, for instance, that you want to look at a person's position in the company and what projects they are part of to decide what data they should see, then you can configure your security labels so that each label can include that information. LBAC is flexible enough to let you set up anything from very complicated criteria, to a very simple system where each label represents either a "high" or a "low" level of trust.

Once created, a security label can be associated with individual columns and rows in a table to protect the data held there. Data that is protected by a security label is called *protected data*. A security administrator allows users access to protected data by granting them security labels. When a user tries to access protected data, that user's security label is compared to the security label protecting the data. The protecting label will block some security labels and not block others.

A user is allowed to hold security labels for multiple security policies at once. For any given security policy, however, a user can hold at most one label for read access and one label for write access.

A security administrator can also grant exemptions to users. An *exemption* allows you to access protected data that your security labels might otherwise prevent you from accessing. Together your security labels and exemptions are called your *LBAC credentials*.

If you try to access a protected column that your LBAC credentials do not allow you to access then the access will fail and you will get an error message.

If you try to read protected rows that your LBAC credentials do not allow you to read then DB2 acts as if those rows do not exist. Those rows cannot be selected as part of any SQL statement that you run, including SELECT, UPDATE, or DELETE. Even the aggregate functions ignore rows that your LBAC credentials do not allow you to read. The COUNT(*) function, for example, will return a count only of the rows that you have read access to.

Views and LBAC:

You can define a view on a protected table the same way you can define one on a non-protected table. When such a view is accessed the LBAC protection on the underlying table is enforced. The LBAC credentials used are those of the session authorization ID. Two users accessing the same view might see different rows depending on their LBAC credentials.

Referential integrity constraints and LBAC:

The following rules explain how LBAC rules are enforced in the presence of referential integrity constraints:

- **Rule 1:** The LBAC read access rules are NOT applied for internally generated scans of child tables. This is to avoid having orphan children.
- **Rule 2:** The LBAC read access rules are NOT applied for internally generated scans of parent tables
- **Rule 3:** The LBAC write rules are applied when a CASCADE operation is performed on child tables. For example, If a user deletes a parent, but cannot

delete any of the children because of an LBAC write rule violation, then the delete should be rolled-back and an error raised.

What LBAC does not do:

- LBAC will never allow access to data that is forbidden by discretionary access control.

Example: If you do not have permission to read from a table then you will not be allowed to read data from that table—even the rows and columns to which LBAC would otherwise allow you access.

- Your LBAC credentials only limit your access to protected data. They have no effect on your access to unprotected data.
- LBAC credentials are not checked when you drop a table or a database, even if the table or database contains protected data.
- LBAC credentials are not checked when you back up your data. If you can run a backup on a table, which rows are backed up is not limited in any way by the LBAC protection on the data. Also, data on the backup media is not protected by LBAC. Only data in the database is protected.
- LBAC cannot be used to protect any of the following types of tables:
 - A materialized query table (MQT)
 - A table that a materialized query table (MQT) depends on
 - A staging table
 - A table that a staging table depends on
 - A typed table
- LBAC protection cannot be applied to a nickname.

LBAC tutorial:

A tutorial leading you through the basics of using LBAC is available online. The tutorial is part of the IBM developerWorks website (<http://www.ibm.com/developerworks/db2>) and is called DB2 Label-Based Access Control, a practical guide.

Related concepts:

- “Database authorities” on page 511
- “LBAC security label components overview” on page 541
- “LBAC security labels” on page 547
- “LBAC security policies” on page 540

LBAC security policies

A *security policy* is a database object that is part of label-based access control (LBAC). A security policy includes this information:

- What security label components will be used in the security labels that are part of the policy
- What rules will be used when comparing those security label components
- Which of certain optional behaviors will be used when accessing data protected by the policy

Every protected table must have one and only one security policy associated with it. Rows and columns in that table can only be protected with security labels that

are part of that security policy and all access of protected data follows the rules of that policy. You can have multiple security policies in a single database but you cannot have more than one security policy protecting any given table.

Creating a security policy:

You must be a security administrator to create a security policy. You create a security policy with the SQL statement `CREATE SECURITY POLICY`. The security label components listed in a security policy must be created before the `CREATE SECURITY POLICY` statement is executed. The order in which the components are listed when a security policy is created does not indicate any sort of precedence or other relationship among the components but it is important to know the order when creating security labels with built-in functions like `SECLABEL`.

Altering a security policy:

Security policies cannot be altered. The only way to change a security policy is to drop it and re-create it.

Dropping a security policy:

You must be a security administrator to drop a security policy. You drop a security policy using the SQL statement `DROP`.

You cannot drop a security policy if it is associated with (added to) any table.

Related concepts:

- “Label-based access control (LBAC) overview” on page 538

Related reference:

- “`CREATE SECURITY LABEL COMPONENT` statement” in *SQL Reference, Volume 2*
- “`CREATE SECURITY POLICY` statement” in *SQL Reference, Volume 2*
- “`DROP` statement” in *SQL Reference, Volume 2*

LBAC security label components

This section explains what you need to know in order to use LBAC security label components.

LBAC security label components overview

A *security label component* is a database object that is part of label-based access control (LBAC). You use security label components to model your organization’s security structure.

A component can represent any criteria that you might use to decide if a user should have access to a given piece of data. Typical examples of such criteria include:

- How well trusted the user is
- What department the user is in
- Whether the user is involved in a particular project

Example: If you want the department that a user is in to affect which data they can access, you could create a component named `dept` and define

elements for that component that name the various departments in your company. You would then include the component dept in your security policy.

An *element* of a security label component is one particular "setting" that is allowed for that component.

Example: A security label component that represents a level of trust might have the four elements: Top Secret, Secret, Classified, and Unclassified.

Creating a security label component:

You must be a security administrator to create a security label component. You create security label components with the SQL statement CREATE SECURITY LABEL COMPONENT.

When you create a security label component you must provide:

- A name for the component
- What type of component it is (ARRAY, TREE, or SET)
- A complete list of allowed elements
- For types ARRAY and TREE you must describe how each element fits into the structure of the component

Types of components:

There are three types of security label components:

- TREE: Each element represents a node in a tree structure
- ARRAY: Each element represents a point on a linear scale
- SET: Each element represents one member of a set

The types are used to model the different ways in which elements can relate to each other. For example, if you are creating a component to describe one or more departments in a company you would probably want to use a component type of TREE because most business structures are in the form of a tree. If you are creating a component to represent the level of trust that a person has, you would probably use a component of type ARRAY because for any two levels of trust, one will always be higher than the other.

The details of each type, including detailed descriptions of the relationships that the elements can have with each other, are described in their own section.

Altering security label components:

Security label components cannot be altered. The only way to change a security label component is to drop it and re-create it.

Dropping a security label component:

You must be a security administrator to drop a security label component. You drop a security label component with the SQL statement DROP.

Related concepts:

- "LBAC security label component type: ARRAY" on page 543
- "LBAC security label component type: SET" on page 543

- “LBAC security label component type: TREE” on page 544

Related reference:

- “CREATE SECURITY LABEL COMPONENT statement” in *SQL Reference, Volume 2*
- “DROP statement” in *SQL Reference, Volume 2*

LBAC security label component type: SET

SET is one type of security label component that can be used in a label-based access control (LBAC) security policy. Components of this type are unordered lists of elements. The only comparison that can be made for elements of this type of component is whether or not a given element is in the list.

Related concepts:

- “Label-based access control (LBAC) overview” on page 538
- “LBAC security label components overview” on page 541
- “LBAC security policies” on page 540

Related reference:

- “CREATE SECURITY LABEL COMPONENT statement” in *SQL Reference, Volume 2*

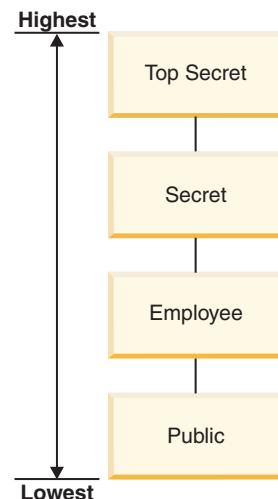
LBAC security label component type: ARRAY

ARRAY is one type of security label component. In this type of component the order in which the elements are listed when the component is created defines a scale with the first element listed being the highest value and the last being the lowest.

Example: If the component mycomp is defined in this way:

```
CREATE SECURITY LABEL COMPONENT mycomp
  ARRAY [ 'Top Secret', 'Secret', 'Employee', 'Public' ]
```

Then the elements are treated as if they are organized in a structure like this:



In a component of type ARRAY, the elements can have these sorts of relationships to each other:

Higher than

Element A is higher than element B if element A is listed earlier in the ARRAY clause than element B.

Lower than

Element A is lower than element B if element A is listed later in the ARRAY clause than element B

Related concepts:

- “Label-based access control (LBAC) overview” on page 538
- “LBAC security label components overview” on page 541
- “LBAC security policies” on page 540

Related reference:

- “CREATE SECURITY LABEL COMPONENT statement” in *SQL Reference, Volume 2*

LBAC security label component type: TREE

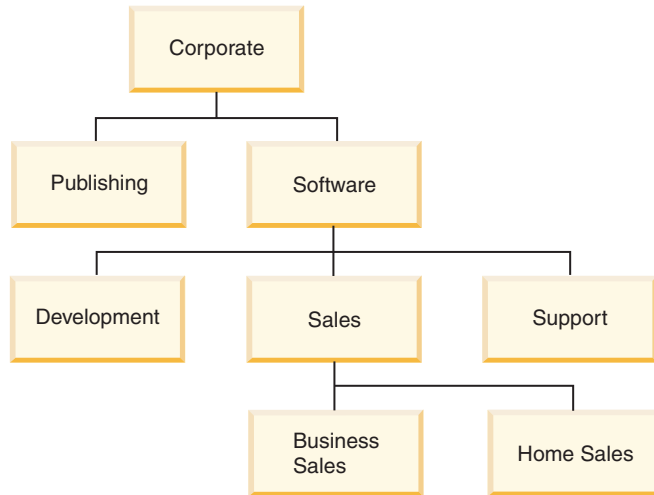
Components of type TREE:

TREE is one type of security label component that can be used in a label-based access control (LBAC) security policy. In this type of component the elements are treated as if they are arranged in a tree structure. When you specify an element that is part of a component of type TREE you must also specify which other element it is under. The one exception is the first element which must be specified as being the ROOT of the tree. This allows you to organize the elements in a tree structure.

Example: If the component mycomp is defined this way:

```
CREATE SECURITY LABEL COMPONENT mycomp
TREE (
    'Corporate'      ROOT,
    'Publishing'    UNDER 'Corporate',
    'Software'      UNDER 'Corporate',
    'Development'  UNDER 'Software',
    'Sales'         UNDER 'Software',
    'Support'      UNDER 'Software'
    'Business Sales' UNDER 'Sales'
    'Home Sales'   UNDER 'Sales'
)
```

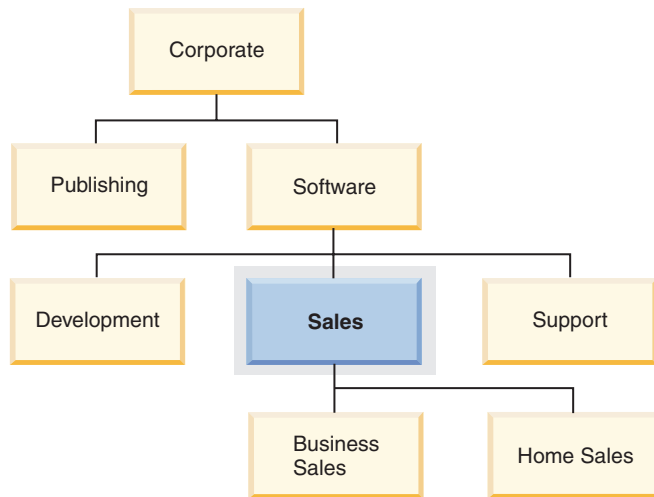
Then the elements are treated as if they are organized in a tree structure like this:



In a component of type TREE, the elements can have these types of relationships to each other:

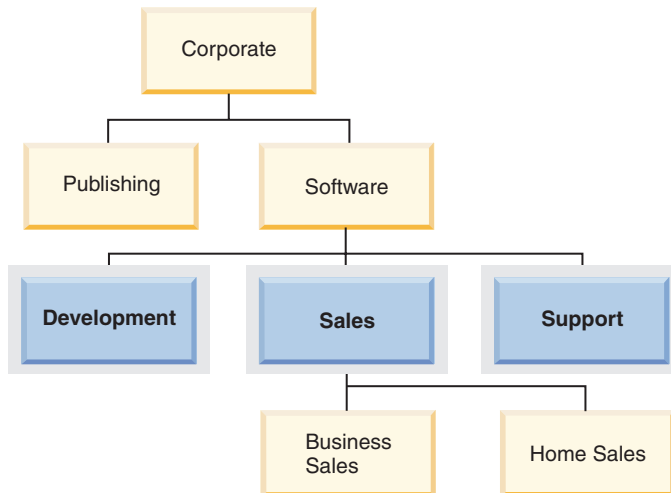
Parent Element A is a parent of element B if element B is UNDER element A.

Example: This diagram shows the parent of the Business Sales element:



Child Element A is a child of element B if element A is UNDER element B.

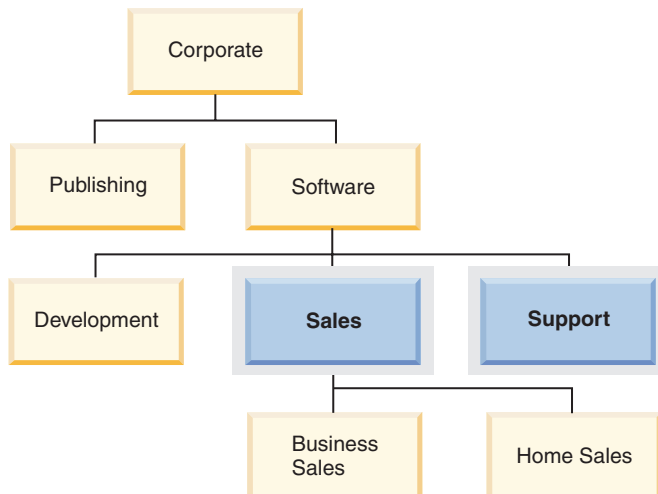
Example: This diagram shows the children of the Software element:



Sibling

Two elements are siblings of each other if they have the same parent.

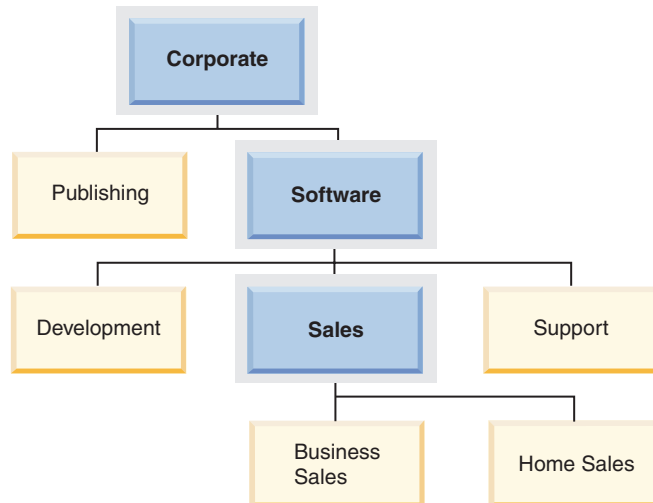
Example: This diagram shows the siblings of the Development element:



Ancestor

Element A is an ancestor of element B if it is the parent of B, or if it is the parent of the parent of B, and so on. The root element is an ancestor of all other elements in the tree.

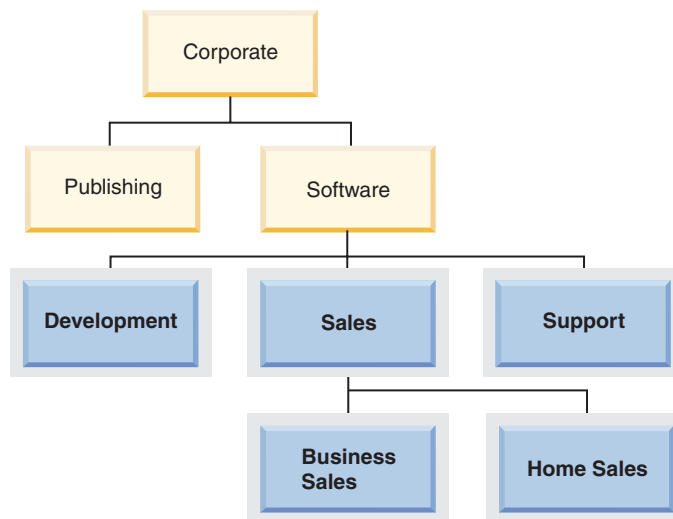
Example: This diagram shows the ancestors of the Home Sales element:



Descendent

Element A is a descendent of element B if it is the child of B, or if it is the child of a child of B, and so on.

Example: This diagram shows the descendents of the Software element:



Related concepts:

- “Label-based access control (LBAC) overview” on page 538
- “LBAC security label components overview” on page 541
- “LBAC security policies” on page 540

Related reference:

- “CREATE SECURITY LABEL COMPONENT statement” in *SQL Reference, Volume 2*

LBAC security labels

In label-based access control (LBAC) a *security label* is a database object that describes a certain set of security criteria. Security labels are applied to data in

order to protect the data. They are granted to users to allow them to access protected data. When a user tries to access protected data, their security label is compared to the security label that is protecting the data. The protecting security label will block some security labels and not block others. If a user's security label is blocked then the user cannot access the data.

Every security label is part of exactly one security policy and includes one value for each component in that security policy. A *value* in the context of a security label component is a list of zero or more of the elements allowed by that component. Values for ARRAY type components can contain zero or one element, values for other types can have zero or more elements. A value that does not include any elements is called an *empty value*.

Example: If a TREE type component has the three elements Human Resources, Sales, and Shipping then these are some of the valid values for that component:

- Human Resources (or any of the elements by itself)
- Human Resources, Shipping (or any other combination of the elements as long as no element is included more than once)
- *An empty value*

Whether a particular security label will block another is determined by the values of each component in the labels and the LBAC rule set that is specified in the security policy of the table. The details of how the comparison is made are given in the section How LBAC security labels are compared.

When security labels are converted to a text string they use the format described in the section Format for security label values.

Creating security labels:

You must be a security administrator to create a security label. You create a security label with the SQL statement CREATE SECURITY LABEL. When you create a security label you provide:

- A name for the label
- The security policy that the label is part of
- Values for one or more of the components included in the security policy

Any components for which a value is not specified is assumed to have an empty value. A security label must have at least one non-empty value.

Altering security labels:

Security labels cannot be altered. The only way to change a security label is to drop it and re-create it.

Dropping security labels:

You must be a security administrator to drop a security label. You drop a security label with the SQL statement DROP. You cannot drop a security label that is being used to protect data anywhere in the database or that is currently held by one or more users.

Granting security labels:

You must be a security administrator to grant a security label to a user. You grant a security label to a user with the SQL statement GRANT SECURITY LABEL. When you grant a security label you can grant it for read access, for write access, or for both read and write access. A user cannot hold two security labels from the same security policy for the same type of access.

Revoking security labels:

You must be a security administrator to revoke a security label from a user. To revoke a security label, use the SQL statement REVOKE SECURITY LABEL.

Data types compatible with security labels:

Security labels have a data type of SYSPROC.DB2SECURITYLABEL. Data conversion is supported between SYSPROC.DB2SECURITYLABEL and VARCHAR(128) FOR BIT DATA.

Related concepts:

- “How LBAC security labels are compared” on page 550
- “Label-based access control (LBAC) overview” on page 538
- “LBAC security label components overview” on page 541
- “Protection of data using LBAC” on page 558

Related reference:

- “CREATE SECURITY LABEL statement” in *SQL Reference, Volume 2*
- “Format for security label values” on page 549

Format for security label values

Sometimes the values in a security label are represented in the form of a character string, for example when using the built-in function SECLABEL. When representing the values in a security label as a string this format is used.

- The values of the components are listed from left to right in the same order that the components are listed in the CREATE SECURITY POLICY statement for the security policy
- An element is represented by the name of that element
- Elements for different components are separated by a colon (:)
- If more than one element are given for the same component the elements are enclosed in parentheses (()) and are separated by a comma (,)
- Empty values are represented by a set of empty parentheses (())

Example: A security label is part of a security policy that has these three components in this order: Level, Department, and Projects. The security label has these values:

Table 29.

Component	Values
Level	Secret
Department	Empty value

Table 29. (continued)

Component	Values
Projects	<ul style="list-style-type: none">• Epsilon 37• Megaphone• Cloverleaf

This security label values look like this as a string:

```
'Secret:():(Epsilon 37,Megaphone,Cloverleaf)'
```

Related concepts:

- “How LBAC security labels are compared” on page 550
- “LBAC security label components overview” on page 541
- “LBAC security labels” on page 547

How LBAC security labels are compared

When you try to access data protected by label-based access control (LBAC), Your LBAC credentials are compared to one or more security labels to see if the access is blocked. Your LBAC credentials are any security labels you hold plus any exemptions that you hold.

There are only two types of comparison that can be made. Your LBAC credentials can be compared to a single security label for read access or your LBAC credentials compared to a single security label for write access. Updating and deleting are treated as being a read followed by a write. When an operation requires multiple comparisons to be made, each is made separately.

Which of your security labels is used:

Even though you might hold multiple security labels only one is compared to the protecting security label. The label used is the one that meets these criteria:

- It is part of the security policy that is protecting the table being accessed.
- It was granted for the type of access (read or write).

If you do not have a security label that meets these criteria then a default security label is assumed that has empty values for all components.

How the comparison is made:

Security labels are compared component by component. If a security label does not have a value for one of the components then an empty value is assumed. As each component is examined, the appropriate rules of the LBAC rule set are used to decide if the elements in your value for that component should be blocked by the elements in the value for the same component in the protecting label. If any of your values are blocked then your LBAC credentials are blocked by the protecting security label.

The LBAC rule set used in the comparison is designated in the security policy. To find out what the rules are and when each one is used, see the description of that rule set.

How exemptions affect comparisons:

If you hold an exemption for the rule that is being used to compare two values then that comparison is not done and the protecting value is assumed not to block the value in your security label.

Example: The LBAC rule set is DB2LBACRULES and the security policy has two components. One component is of type ARRAY and the other is of type TREE. The user has been granted an exemption on the rule DB2LBACREADTREE, which is the rule used for read access when comparing values of components of type TREE. If the user attempts to read protected data then whatever value the user has for the TREE component, even if it is an empty value, will not block access because that rule is not used. Whether the user can read the data depends entirely on the values of the ARRAY component of the labels.

Related concepts:

- “Label-based access control (LBAC) overview” on page 538
- “LBAC rule exemptions” on page 556
- “LBAC rule set: DB2LBACRULES” on page 552
- “LBAC rule sets overview” on page 551
- “LBAC security labels” on page 547
- “LBAC security policies” on page 540

LBAC rule sets

This section explains LBAC rule sets in general and also the details of the DB2LBACRULES rule set.

LBAC rule sets overview

An LBAC rule set is a predefined set of rules that are used when comparing security labels. When the values of a two security labels are being compared, one or more of the rules in the rule set will be used to determine if one value blocks another.

Each LBAC rule set is identified by a unique name. When you create a security policy you must specify the LBAC rule set that will be used with that policy. Any comparison of security labels that are part of that policy will use that LBAC rule set.

Each rule in a rule set is also identified by a unique name. You use the name of a rule when you are granting an exemption on that rule.

How many rules are in a set and when each rule is used can vary from rule set to rule set.

There is currently only one supported LBAC rule set. The name of that rule set is DB2LBACRULES.

Related concepts:

- “Label-based access control (LBAC) overview” on page 538
- “LBAC rule set: DB2LBACRULES” on page 552

LBAC rule set: DB2LBACRULES

This LBAC rule set provides a traditional set of rules for comparing the values of security label components. It protects from both write-up and write-down.

What are write-up and write down?:

Write-up and write-down apply only to components of type ARRAY and only to write access. Write up occurs when the value protecting data that you are writing to is higher than your value. Write-down is when the value protecting the data is lower than yours. By default neither write-up nor write-down is allowed, meaning that you can only write data that is protected by the same value that you have.

When comparing two values for the same component, which rules are used depends on the type of the component (ARRAY, SET, or TREE) and what type of access is being attempted (read, or write). This table lists the rules, tells when each is used, and describes how the rule determines if access is blocked.

Table 30. Summary of the DB2LBACRULES rules

Rule name	Used when comparing the values of this type of component	Used when attempting this type of access	Access is blocked when this condition is met
DB2LBACREADARRAY	ARRAY	Read	The user's value is lower than the protecting value.
DB2LBACREADSET	SET	Read	There are one or more protecting values that the user does not hold.
DB2LBACREADTREE	TREE	Read	None of the user's values is equal to or an ancestor of one of the protecting values.
DB2LBACWRITEARRAY	ARRAY	Write	The user's value is higher than the protecting value or lower than the protecting value. ¹
DB2LBACWRITASET	SET	Write	There are one or more protecting values that the user does not hold.
DB2LBACWRITETREE	TREE	Write	None of the user's values is equal to or an ancestor of one of the protecting values.

Notes:

1. The DB2LBACWRITEARRAY rule can be thought of as being two different rules combined. One prevents writing to data that is higher than your level (write-up) and the other prevents writing to data that is lower than your level (write-down). When granting an exemption to this rule you can exempt the user from either of these rules or from both.

How the rules handle empty values:

All rules treat empty values the same way. An empty value blocks no other values and is blocked by any non-empty value.

Examples:

DB2LBACREADSET and DB2LBACWRITASET examples:

These examples are valid for a user trying to read or trying to write protected data. They assume that the values are for a component of

type SET that has these elements: one two three four

Table 31. Examples of applying the DB2LBACREADSET and DB2LBACWRITESET rules.

User's value	Protecting value	Access blocked?
'one'	'one'	Not blocked. The values are the same.
'(one,two,three)'	'one'	Not blocked. The user's value contains the element 'one'.
'(one,two)'	'(one,two,four)'	Blocked. The element 'four' is in the protecting value but not in the user's value.
'()'	'one'	Blocked. An empty value is blocked by any non-empty value.
'one'	'()'	Not blocked. No value is blocked by an empty value.
'()'	'()'	Not blocked. No value is blocked by an empty value.

DB2LBACREADTREE and DB2LBACWRITETREE:

These examples are valid for both read access and write access. They assume that the values are for a component of type TREE that was defined in this way:

```
CREATE SECURITY LABEL COMPONENT mycomp
TREE (
  'Corporate'      ROOT,
  'Publishing'    UNDER 'Corporate',
  'Software'      UNDER 'Corporate',
  'Development'   UNDER 'Software',
  'Sales'         UNDER 'Software',
  'Support'       UNDER 'Software',
  'Business Sales' UNDER 'Sales',
  'Home Sales'    UNDER 'Sales'
)
```

This means the elements are in this arrangement:

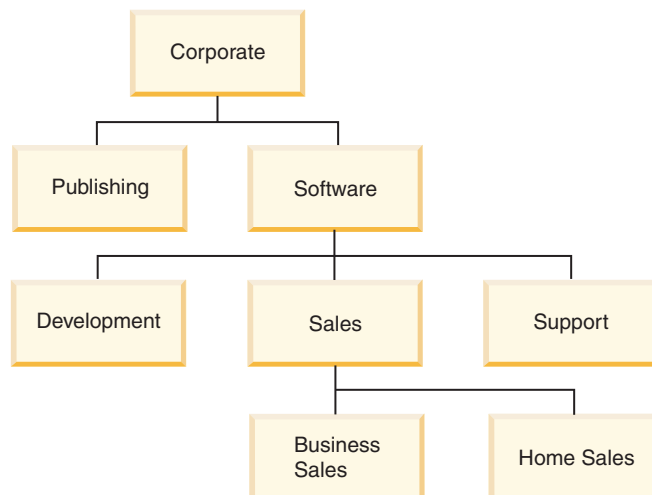


Table 32. Examples of applying the DB2LBACREADTREE and DB2LBACWRITETREE rules.

User's value	Protecting value	Access blocked?
'(Support,Sales)'	'Development'	Blocked. The element 'Development' is not one of the user's values and neither 'Support' nor 'Sales' is an ancestor of 'Development'.
'(Development,Software)'	'(Business Sales,Publishing)'	Not blocked. The element 'Software' is an ancestor of 'Business Sales'.
'(Publishing,Sales)'	'(Publishing,Support)'	Not blocked. The element 'Publishing' is in both sets of values.
'Corporate'	'Development'	Not blocked. The root value is an ancestor of all other values.
'()'	'Sales'	Blocked. An empty value is blocked by any non-empty value.
'Home Sales'	'()'	Not blocked. No value is blocked by an empty value.
'()'	'()'	Not blocked. No value is blocked by an empty value.

DB2LBACREADARRAY examples:

These examples are for read access only. They assume that the values are for a component of type ARRAY that includes these elements in this arrangement:

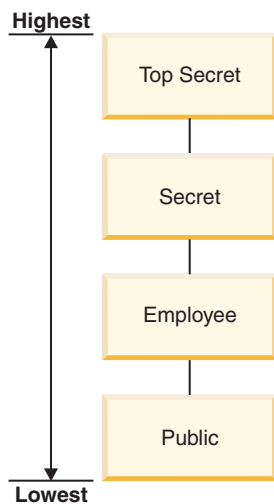


Table 33. Examples of applying the DB2LBACREADARRAY rule.

User's value	Protecting value	Read access blocked?
'Secret'	'Employee'	Not blocked. The element 'Secret' is higher than the element 'Employee'.
'Secret'	'Secret'	Not blocked. The values are the same.

Table 33. Examples of applying the DB2LBACREADARRAY rule. (continued)

User's value	Protecting value	Read access blocked?
'Secret'	'Top Secret'	Blocked. The element 'Top Secret' is higher than the element 'Secret'.
('')	'Public'	Blocked. An empty value is blocked by any non-empty value.
'Public'	('')	Not blocked. No value is blocked by an empty value.
('')	('')	Not blocked. No value is blocked by an empty value.

DB2LBACWRITEARRAY examples:

These examples are for write access only. They assume that the values are for a component of type ARRAY that includes these elements in this arrangement:

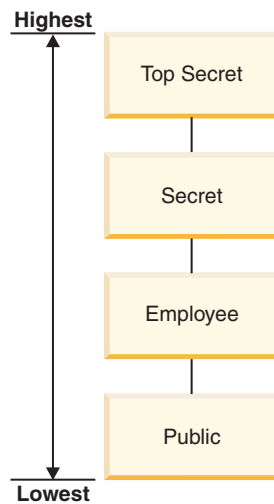


Table 34. Examples of applying the DB2LBACWRITEARRAY rule.

User's value	Protecting value	Write access blocked?
'Secret'	'Employee'	Blocked. The element 'Employee' is lower than the element 'Secret'.
'Secret'	'Secret'	Not blocked. The values are the same.
'Secret'	'Top Secret'	Blocked. The element 'Top Secret' is higher than the element 'Secret'.
('')	'Public'	Blocked. An empty value is blocked by any non-empty value.
'Public'	('')	Not blocked. No value is blocked by an empty value.
('')	('')	Not blocked. No value is blocked by an empty value.

Related concepts:

- "How LBAC security labels are compared" on page 550

- “LBAC rule sets overview” on page 551

LBAC rule exemptions

Exemptions:

An LBAC rule exemption is part of the label-based access control (LBAC) feature. When you hold an exemption on a particular rule of a particular security policy that rule is not enforced when you try to access data protected by that security policy. An exemption has no effect when comparing security labels of any security policy other than the one for which it was granted.

Example:

There are two tables: T1 and T2. T1 is protected by security policy P1 and T2 is protected by security policy P2. Both security policies have one component. The component of each is of type ARRAY. T1 and T2 each contain only one row of data. The security label that you hold for read access under security policy P1 does not allow you access to the row in T1. The security label that you hold for read access under security policy P2 does not allow you read access to the row in T2.

Now you are granted an exemption on DB2LBACREADARRAY under P1. You can now read the row from T1 but not the row from T2 because T2 is protected by a different security policy and you do not hold an exemption to the DB2LBACREADARRAY rule in that policy.

You can hold multiple exemptions. If you hold an exemption to every rule used by a security policy then you will have complete access to all data protected by that security policy.

Granting LBAC rule exemptions:

You must have security administrator (SECADM) authority to grant an LBAC rule exemption. To grant an LBAC rule exemption, use the SQL statement `GRANT EXEMPTION ON RULE`.

When you grant an LBAC rule exemption you provide this information:

- The rule or rules that the exemption is for
- The security policy that the exemption is for
- The user to which you are granting the exemption

Important: LBAC rule exemptions provide very powerful access. Do not grant them without careful consideration.

Revoking LBAC rule exemptions:

You must have security administrator (SECADM) authority to revoke an LBAC rule exemption. To revoke an LBAC rule exemption, use the SQL statement `REVOKE EXEMPTION ON RULE`.

Related concepts:

- “How LBAC security labels are compared” on page 550
- “LBAC rule sets overview” on page 551

- “LBAC security policies” on page 540

Related reference:

- “GRANT (Exemption) statement” in *SQL Reference, Volume 2*
- “REVOKE (Exemption) statement” in *SQL Reference, Volume 2*

Built-in functions for dealing with LBAC security labels

Three built-in functions are provided for dealing with label-based access control (LBAC) security labels. Each is described briefly here and in detail in the *SQL Reference*

SECLABEL:

This built-in function is used to build a security label by specifying a security policy and values for each of the components in the label. The returned value has a data type of DB2SECURITYLABEL and is a security label that is part of the indicated security policy and has the indicated values for the components. It is not necessary that a security label with the indicated values already exists.

Example: Table T1 has two columns, the first has a data type of DB2SECURITYLABEL and the second has a data type of INTEGER. T1 is protected by security policy P1, which has three security label components: level, departments, and groups. If UNCLASSIFIED is an element of the component level, ALPHA and SIGMA are both elements of the component departments, and G2 is an element of the component groups then a security label could be inserted like this:

```
INSERT INTO T1 VALUES ( SECLABEL( 'P1', 'UNCLASSIFIED:(ALPHA,SIGMA):G2' ), 22 )
```

SECLABEL_BY_NAME:

This built-in function accepts the name of a security policy and the name of a security label that is part of that security policy. It then returns the indicated security label as a DB2SECURITYLABEL. You must use this function when inserting an existing security label into a column that has a data type of DB2SECURITYLABEL.

Example: Table T1 has two columns, the first has a data type of DB2SECURITYLABEL and the second has a data type of INTEGER. The security label named L1 is part of security policy P1. This SQL inserts the security label:

```
INSERT INTO T1 VALUES ( SECLABEL_BY_NAME( 'P1', 'L1' ), 22 )
```

This SQL does not work:

```
INSERT INTO T1 VALUES ( P1.L1, 22 ) // Syntax Error!
```

SECLABEL_TO_CHAR:

This built-in function returns a string representation of the values that make up a security label.

Example: Column C1 in table T1 has a data type of DB2SECURITYLABEL. T1 is protected by security policy P1, which has three security label components: level, departments, and groups. There is one row in T1 and the value in column C1 that has these elements for each of the

components:

Component	Elements
level	SECRET
departments	DELTA and SIGMA
groups	G3

A user that has LBAC credentials that allow reading the row executes this SQL statement:

```
SELECT SECLABEL_TO_CHAR( 'P1', C1 ) AS C1 FROM T1
```

The output looks like this:

C1

```
'SECRET:(DELTA,SIGMA):G3'
```

Related concepts:

- “Label-based access control (LBAC) overview” on page 538
- “LBAC security labels” on page 547

Related reference:

- “SECLABEL_BY_NAME scalar function” in *SQL Reference, Volume 1*
- “SECLABEL_TO_CHAR scalar function” in *SQL Reference, Volume 1*
- “SECLABEL scalar function” in *SQL Reference, Volume 1*
- “Format for security label values” on page 549

Protection of data using LBAC

Protecting tables:

Label-based access control (LBAC) can be used to protect rows of data, columns of data, or both. Data in a table can only be protected by security labels that are part of the security policy protecting the table. Data protection, including adding a security policy, can be done when creating the table or later by altering the table. You can add a security policy to a table and protect data in that table as part of the same CREATE TABLE or ALTER TABLE statement.

As a general rule you are not allowed to protect data in such a way that your current LBAC credentials do not allow you to write to that data.

Adding a security policy to a table:

You can add a security policy to a table when you create the table by using the SECURITY POLICY clause of the CREATE TABLE statement. You can add a security policy to an existing table by using the ADD SECURITY POLICY clause of the ALTER TABLE statement. You do not need to have SECADM authority or have LBAC credentials to add a security policy to a table.

Security policies cannot be added to types of tables that cannot be protected by LBAC. See the overview of LBAC for a list of table types that cannot be protected by LBAC.

No more than one security policy can be added to any table.

Protecting rows:

You can allow protected rows in a new table by including a column with a data type of DB2SECURITYLABEL when you create the table. The CREATE TABLE statement must also add a security policy to the table. You do not need to have SECADM authority or have any LBAC credentials to create such a table.

You can allow protected rows in an existing table by adding a column that has a data type of DB2SECURITYLABEL. To add such a column, either the table must already be protected by a security policy or the ALTER TABLE statement that adds the column must also add a security policy to the table. When the column is added, the security label you hold for write access is used to protect all existing rows. If you do not hold a security label for write access that is part of the security policy protecting the table then you cannot add a column that has a data type of DB2SECURITYLABEL.

After a table has a column of type DB2SECURITYLABEL you protect each new row of data by storing a security label in that column. The details of how this works are described in the topics about inserting and updating LBAC protected data. You must have LBAC credentials to insert rows into a table that has a column of type DB2SECURITYLABEL.

A column that has a data type of DB2SECURITYLABEL cannot be dropped and cannot be changed to any other data type.

Protecting columns:

You can protect a column when you create the table by using the SECURED WITH column option of the CREATE TABLE statement. You can add protection to an existing column by using the SECURED WITH option in an ALTER TABLE statement.

To protect a column with a particular security label you must have LBAC credentials that allow you to write to data protected by that security label. You do not have to have SECADM authority.

Columns can only be protected by security labels that are part of the security policy protecting the table. You cannot protect columns in a table that has no security policy. You are allowed to protect a table with a security policy and protect one or more columns in the same statement.

You can protect any number of the columns in a table but a column can be protected by no more than one security label.

Related concepts:

- “Inserting of LBAC protected data” on page 563
- “Label-based access control (LBAC) overview” on page 538
- “LBAC security labels” on page 547
- “LBAC security policies” on page 540
- “Removal of LBAC protection from data” on page 572
- “Updating of LBAC protected data” on page 565

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Reading of LBAC protected data

When you try to read data protected by label-based access control (LBAC), your LBAC credentials for reading are compared to the security label that is protecting the data. If the protecting label does not block your credentials you are allowed to read the data.

In the case of a protected column the protecting security label is defined in the schema of the table. The protecting security label for that column is the same for every row in the table. In the case of a protected row the protecting security label is stored in the row in a column of type DB2SECURITYLABEL. It can be different for every row in the table.

The details of how your LBAC credentials are compared to a security label are given in How LBAC security labels are compared.

Reading protected columns:

When you try to read from a protected column your LBAC credentials are compared with the security label protecting the column. Based on this comparison access will either be blocked or allowed. If access is blocked then an error is returned and the statement fails. Otherwise, the statement proceeds as usual.

Trying to read a column that your LBAC credentials do not allow you to read, causes the entire statement to fail.

Example:

Table T1 has two protected columns. The column C1 is protected by the security label L1. The column C2 is protected by the security label L2.

Assume that user Jyoti has LBAC credentials for reading that allow access to security label L1 but not to L2. If Jyoti issues the following SQL statement, the statement will fail:

```
SELECT * FROM T1
```

The statement fails because column C2 is included in the SELECT clause as part of the wildcard (*).

If Jyoti issues the following SQL statement it will succeed:

```
SELECT C1 FROM T1
```

The only protected column in the SELECT clause is C1, and Jyoti's LBAC credentials allow her to read that column.

Reading protected rows:

If you do not have LBAC credentials that allow you to read a row it is as if that row does not exist for you.

When you read protected rows, only those rows to which your LBAC credentials allow read access are returned. This is true even if the column of type DB2SECURITYLABEL is not part of the SELECT clause.

Depending on their LBAC credentials, different users might see different rows in a table that has protected rows. For example, two users executing the statement `SELECT COUNT(*) FROM T1` may get different results if T1 has protected rows and the users have different LBAC credentials.

Your LBAC credentials affect not only SELECT statements but also other SQL statements like UPDATE, and DELETE. If you do not have LBAC credentials that allow you to read a row, you cannot affect that row.

Example:

Table T1 has these rows and columns. The column ROWSECURITYLABEL has a data type of DB2SECURITYLABEL.

Table 35.

LASTNAME	DEPTNO	ROWSECURITYLABEL
Rjaibi	55	L2
Miller	77	L1
Fielding	11	L3
Bird	55	L2

Assume that user Dan has LBAC credentials that allow him to read data that is protected by security label L1 but not data protected by L2 or L3.

Dan issues the following SQL statement:

```
SELECT * FROM T1
```

The SELECT statement returns only the row for Miller. No error messages or warning are returned.

Dan's view of table T1 is this:

Table 36.

LASTNAME	DEPTNO	ROWSECURITYLABEL
Miller	77	L1

The rows for Rjaibi, Fielding, and Bird are not returned because read access is blocked by their security labels. Dan cannot delete or update these rows. They will also not be included in any aggregate functions. For Dan it is as if those rows do not exist.

Dan issues this SQL statement:

```
SELECT COUNT(*) FROM T1
```

The statement returns a value of 1 because only the row for Miller can be read by the user Dan.

Reading protected rows that contain protected columns:

Column access is checked before row access. If your LBAC credentials for read access are blocked by the security label protecting one of the columns you are selecting then the entire statement fails. If not, the statement continues and only the rows protected by security labels to which your LBAC credentials allow read access are returned.

Example:

The column LASTNAME of table T1 is protected with the security label L1. The column DEPTNO is protected with security label L2. The column ROWSECURITYLABEL has a data type of DB2SECURITYLABEL. T1, including the data, looks like this:

Table 37.

LASTNAME <i>Protected by L1</i>	DEPTNO <i>Protected by L2</i>	ROWSECURITYLABEL
Rjaibi	55	L2
Miller	77	L1
Fielding	11	L3

Assume that user Sakari has LBAC credentials that allow reading data protected by security label L1 but not L2 or L3.

Sakari issues this SQL statement:

```
SELECT * FROM T1
```

The statement fails because the SELECT clause uses the wildcard (*) which includes the column DEPTNO. The column DEPTNO is protected by security label L2, which Sakari's LBAC credentials do not allow her to read.

Sakari next issues this SQL statement:

```
SELECT LASTNAME, ROWSECURITYLABEL FROM T1
```

The select clause does not include any columns that Sakari is not able to read so the statement continues. Only one row is returned, however, because each of the other rows is protected by security label L2 or L3.

Table 38.

LASTNAME	ROWSECURITYLABEL
Miller	L1

Related concepts:

- “Deleting or dropping of LBAC protected data” on page 569
- “How LBAC security labels are compared” on page 550
- “Inserting of LBAC protected data” on page 563
- “Label-based access control (LBAC) overview” on page 538
- “LBAC security labels” on page 547
- “Updating of LBAC protected data” on page 565

Inserting of LBAC protected data

Inserting to protected columns:

When you try to explicitly insert data to a protected column your LBAC credentials for writing are compared with the security label protecting that column. Based on this comparison access will either be blocked or allowed.

The details of how two security labels are compared are given in How LBAC security labels are compared.

If access is allowed, the statement proceeds as usual. If access is blocked, then the insert fails and an error is returned.

If you are inserting a row but do not provide a value for a protected column then a default value is inserted if one is available. This happens even if your LBAC credentials do not allow write access to that column. A default is available in the following cases:

- The column was declared with the WITH DEFAULT option
- The column is a generated column
- The column has a default value that is given through a BEFORE trigger
- The column has a data type of DB2SECURITYLABEL, in which case security label that you hold for write access is the default value

Inserting protected rows:

When you insert a new row into a table with protected rows you do not have to provide a value for the column that is of type DB2SECURITYLABEL. If you do not provide a value for that column the column is automatically populated with the security label you have been granted for write access. If you have not been granted a security label for write access an error is returned and the insert fails.

By using built-in functions like SECLABEL you can explicitly provide a security label to be inserted in a column of type DB2SECURITYLABEL. The provided security label is only used, however, if your LBAC credentials would allow you to write to data that is protected with the security label you are trying to insert.

If you provide a security label that you would not be able to write to then what happens depends on the security policy that is protecting the table. If the CREATE SECURITY POLICY statement that created the policy included the option RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL then the insert fails and an error is returned. If the CREATE SECURITY POLICY statement did not include the option or if it instead included the OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL option then the security label you provide is ignored and the security label you hold for write access is used instead. No error or warning is issued in this case.

Examples:

Table T1 is protected by a security policy (named P1) that was created without the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option. Table T1 has two columns but no rows. The columns are LASTNAME and LABEL. The column LABEL has a data type of DB2SECURITYLABEL.

User Joe holds a security label L2 for write access. Assume that the security label L2 allows him to write to data protected by security label L2 but not to data protected by security labels L1 or L3.

Joe issues the following SQL statement:

```
INSERT INTO T1 (LASTNAME, DEPTNO) VALUES ('Rjaibi', 11)
```

Because no security label was included in the INSERT statement, Joe's security label for write access is inserted into the LABEL row.

Table T1 now looks like this:

Table 39.

LASTNAME	LABEL
Rjaibi	L2

Joe issues the following SQL statement, in which he explicitly provides the security label to be inserted into the column LABEL:

```
INSERT INTO T1 VALUES ('Miller', SECLABEL_BY_NAME('P1', 'L1'))
```

The SECLABEL_BY_NAME function in the statement returns a security label that is part of security policy P1 and is named L1. Joe is not allowed to write to data that is protected with L1 so he is not allowed to insert L1 into the column LABEL.

Because the security policy protecting T1 was created without the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option the security label that Joe holds for writing is inserted instead. No error or message is returned.

The table now looks like this:

Table 40.

LASTNAME	LABEL
Rjaibi	L2
Miller	L2

If the security policy protecting the table had been created with the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option then the insert would have failed and an error would have been returned.

Next Joe is granted an exemption to one of the LBAC rules. Assume that his new LBAC credentials allow him to write to data that is protected with security labels L1 and L2. The security label granted to Joe for write access does not change, it is still L2.

Joe issues the following SQL statement:

```
INSERT INTO T1 VALUES ('Bird', SECLABEL_BY_NAME('P1', 'L1'))
```

Because of his new LBAC credentials Joe is able to write to data that is protected by the security label L1. The insertion of L1 is therefore allowed. The table now looks like this:

Table 41.

LASTNAME	LABEL
Rjaibi	L2
Miller	L2
Bird	L1

Related concepts:

- “Deleting or dropping of LBAC protected data” on page 569
- “How LBAC security labels are compared” on page 550
- “Label-based access control (LBAC) overview” on page 538
- “LBAC security labels” on page 547
- “LBAC security policies” on page 540
- “Reading of LBAC protected data” on page 560
- “Updating of LBAC protected data” on page 565

Updating of LBAC protected data

Your LBAC credentials must allow you write access to data before you can update it. In the case of updating a protected row your LBAC credentials must also allow read access to the row.

Updating protected columns:

When you try to update data in a protected column, your LBAC credentials are compared to the security label protecting the column. The comparison made is for write access. If write access is blocked then an error is returned and the statement fails, otherwise the update continues.

The details of how your LBAC credentials are compared to a security label are given in How LBAC security labels are compared.

Example:

Assume there is a table T1 in which column DEPTNO is protected by a security label L2 and column PAYSACLE is protected by a security label L3. T1, including its data, looks like this:

Table 42.

EMPNO	LASTNAME	DEPTNO <i>Protected by</i> L2	PAYSACLE <i>Protected by</i> L3
1	Rjaibi	11	4
2	Miller	11	7
3	Bird	11	9

User Lhakpa has no LBAC credentials. He issues this SQL statement:

```
UPDATE T1 SET EMPNO = 4 WHERE LASTNAME = "Bird"
```

This statement executes without error because it does not update any protected columns. T1 now looks like this:

Table 43.

EMPNO	LASTNAME	DEPTNO <i>Protected by L2</i>	PAYSCALE <i>Protected by L3</i>
1	Rjaibi	11	4
2	Miller	11	7
4	Bird	11	9

Lhakpa next issues this SQL statement:

```
UPDATE T1 SET DEPTNO = 55 WHERE LASTNAME = "Miller"
```

This statement fails and an error is returned because DEPTNO is protected and Lhakpa has no LBAC credentials.

Assume Lhakpa is granted LBAC credentials and that allow the access summarized in the following table. The details of what those credentials are and what elements are in the security labels are not important for this example.

Security label protecting the data	Can read?	Can Write?
L2	No	Yes
L3	No	No

Lhakpa issues this SQL statement again:

```
UPDATE T1 SET DEPTNO = 55 WHERE LASTNAME = "Miller"
```

This time the statement executes without error because Lhakpa's LBAC credentials allow him to write to data protected by the security label that is protecting the column DEPTNO. It does not matter that he is not able to read from that same column. The data in T1 now looks like this:

Table 44.

EMPNO	LASTNAME	DEPTNO <i>Protected by L2</i>	PAYSCALE <i>Protected by L3</i>
1	Rjaibi	11	4
2	Miller	55	7
4	Bird	11	9

Next Lhakpa issues this SQL statement:

```
UPDATE T1 SET DEPTNO = 55, PAYSCALE = 4 WHERE LASTNAME = "Bird"
```

The column PAYSCALE is protected by the security label L3 and Lhakpa's LBAC credentials do not allow him to write to it. Because Lhakpa is unable to write to the column, the update fails and no data is changed.

Updating protected rows:

If your LBAC credentials do not allow you to read a row then it is as if that row does not exist for you so there is no way for you to update that row. For rows that you are able to read, you must also be able to write to the row in order to update it.

When you try to update a row your LBAC credentials for writing are compared to the security label protecting the row. If write access is blocked the update fails and an error is returned. If write access is not blocked then the update continues.

The update that is performed is done the same way as an update to a non-protected row except for the treatment of the column that has a data type of DB2SECURITYLABEL. If you do not explicitly set the value of that column it is automatically set to the security label that you hold for write access. If you do not have a security label for write access an error is returned and the statement fails.

If the update explicitly sets the column that has a data type of DB2SECURITYLABEL then your LBAC credentials are checked again. If the update you are trying to perform would create a row that your current LBAC credentials would not allow you to write to then an error is returned and the statement fails. Otherwise the column is set to the provided security label.

Example:

Assume that table T1 is protected by a security policy named P1 and has a column named LABEL that has a data type of DB2SECURITYLABEL.

T1, including its data, looks like this:

Table 45.

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	11	L1
2	Miller	11	L2
3	Bird	11	L3

Assume that user Jenni has LBAC credentials that allow her to read and write data protected by the security labels L0 and L1 but not data protected by any other security labels. The security label she holds for both read and write is L0. The details of her full credentials and of what elements are in the labels are not important for this example.

Jenni issues this SQL statement:

```
SELECT * FROM T1
```

Jenni sees only one row in the table:

Table 46.

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	11	L1

The rows protected by labels L2 and L3 are not included in the result set because Jenni's LBAC credentials do not allow her to read those rows. For Jenni it is as if those rows do not exist.

Jenni issues these SQL statements:

```
UPDATE T1 SET DEPTNO = 44 WHERE DEPTNO = 11;  
SELECT * FROM T1;
```

The result set returned by the query looks like this:

Table 47.

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L0

The actual data in the table looks like this:

Table 48.

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L0
2	Miller	11	L2
3	Bird	11	L3

The statement executed without error but affected only the first row. The second and third rows are not readable by Jenni so they are not selected for update by the statement even though they meet the condition in the WHERE clause.

Notice that the value of the LABEL column in the updated row has changed even though that column was not explicitly set in the UPDATE statement. The column was set to the security label that Jenni held for writing.

Now Jenni is granted LBAC credentials that allow her to read data protected by any security label. Her LBAC credentials for writing do not change. She is still only able to write to data protected by L0 and L1.

Jenni again issues this SQL statement:

```
UPDATE T1 SET DEPTNO = 44 WHERE DEPTNO = 11
```

This time the update fails because of the second and third rows. Jenni is able to read those rows, so they are selected for update by the statement. She is not, however, able to write to them because they are protected by security labels L2 and L3. The update does not occur and an error is returned.

Jenni now issues this SQL statement:

```
UPDATE T1  
SET DEPTNO = 55, LABEL = SECLABEL_BY_NAME( 'P1', 'L2' )  
WHERE LASTNAME = "Rjaibi"
```

The SECLABEL_BY_NAME function in the statement returns the security label named L2. Jenni is trying to explicitly set the security label protecting the first row. Jenni's LBAC credentials allow her to read the first row, so it is selected for update. Her LBAC credentials allow her to write to rows protected by the security label L0 so she is allowed to update the row. Her LBAC credentials would not, however, allow her to write to a row protected by the security label L2, so she is not allowed

to set the column LABEL to that value. The statement fails and an error is returned. No columns in the row are updated.

Jenni now issues this SQL statement:

```
UPDATE T1 SET LABEL = SECLABEL_BY_NAME( 'P1', 'L1' ) WHERE LASTNAME = "Rjaibi"
```

The statement succeeds because she would be able to write to a row protected by the security label L1.

T1 now looks like this:

Table 49.

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L1
2	Miller	11	L2
3	Bird	11	L3

Updating protected rows that contain protected columns:

If you try to update protected columns in a table with protected rows then your LBAC credentials must allow writing to all of the protected columns affected by the update, otherwise the update fails and an error is returned. This is as described in preceding section **Updating protected columns**. If you are allowed to update all of the protected columns affected by the update you will still only be able to update rows that your LBAC credentials allow you to both read from and write to. This is as described in the preceding section **Updating protected rows**. The handling of a column with a data type of DB2SECURITYLABEL is the same whether the update affects protected columns or not.

If the column that has a data type of DB2SECURITYLABEL is itself a protected column then your LBAC credentials must allow you to write to that column or you cannot update any of the rows in the table.

Related concepts:

- “Deleting or dropping of LBAC protected data” on page 569
- “Inserting of LBAC protected data” on page 563
- “Reading of LBAC protected data” on page 560

Deleting or dropping of LBAC protected data

Deleting protected rows:

If your LBAC credentials do not allow you to read a row then it is as if that row does not exist for you so there is no way for you to delete it.

To delete a row that you are able to read, your LBAC credentials must also allow you to write to the row. When you try to delete a row, your LBAC credentials for writing are compared to the security label protecting the row. If the protecting security label blocks write access by your LBAC credentials, the DELETE statement fails, an error is returned, and no rows are deleted.

Example:

Protected table T1 has these rows:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Miller	77	L1
Bird	55	L2
Fielding	77	L3

Assume that user Pat has LBAC credentials such that her access is as summarized in this table:

Security label	Read access?	Write access?
L1	Yes	Yes
L2	Yes	No
L3	No	No

The exact details of her LBAC credentials and of the security labels are unimportant for this example.

Pat issues the following SQL statement:

```
SELECT * FROM T1 WHERE DEPTNO != 999
```

The statement executes and returns this result set:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Miller	77	L1
Bird	55	L2

The last row of T1 is not included in the results because Pat does not have read access to that row. It is as if that row does not exist for Pat.

Pat issues this SQL statement:

```
DELETE FROM T1 WHERE DEPTNO != 999
```

Pat does not have write access to the first or third row, both of which are protected by L2. So even though she can read the rows she cannot delete them. The DELETE statement fails and no rows are deleted.

Pat issues this SQL statement:

```
DELETE FROM T1 WHERE DEPTNO = 77;
```

This statement succeeds because Pat is able to write to the row with Miller in the LASTNAME column. That is the only row selected by the statement. The row with Fielding in the LASTNAME column is not selected because Pat's LBAC credentials do not allow her to read that row. That row is never considered for the delete so no error occurs.

The actual rows of the table now look like this:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Bird	55	L2
Fielding	77	L3

Deleting rows that have protected columns:

To delete any row in a table that has protected columns you must have LBAC credentials that allow you to write to all protected columns in the table. If there is any row in the table that your LBAC credentials do not allow you to write to then the delete will fail and an error will be returned.

If the table has both protected columns and protected rows then to delete a particular row you must have LBAC credentials that allow you to write to every protected column in the table and also to read from and write to the row that you want to delete.

Example:

In protected table T1, the column DEPTNO is protected by the security label L2. T1 contains these rows:

LASTNAME	DEPTNO <i>Protected by L2</i>	LABEL
Rjaibi	55	L2
Miller	77	L1
Bird	55	L2
Fielding	77	L3

Assume that user Benny has LBAC credentials that allow him the access summarized in this table:

Security label	Read access?	Write access?
L1	Yes	Yes
L2	Yes	No
L3	No	No

The exact details of his LBAC credentials and of the security labels are unimportant for this example.

Benny issues the following SQL statement:

```
DELETE FROM T1 WHERE DEPTNO = 77
```

The statement fails because Benny does not have write access to the column DEPTNO.

Now Benny's LBAC credentials are changed so that he has access as summarized in this table:

Security label	Read access?	Write access?
L1	Yes	Yes
L2	Yes	Yes
L3	Yes	No

Benny issues this SQL statement again:

```
DELETE FROM T1 WHERE DEPTNO = 77
```

This time Benny has write access to the column DEPTNO so the delete continues. The delete statement selects only the row that has a value of Miller in the LASTNAME column. The row that has a value of Fielding in the LASTNAME column is not selected because Benny's LBAC credentials do not allow him to read that row. Because the row is not selected for deletion by the statement it does not matter that Benny is unable to write to the row.

The one row selected is protected by the security label L1. Benny's LBAC credentials allow him to write to data protected by L1 so the delete is successful.

The actual rows in table T1 now look like this:

LASTNAME	DEPTNO <i>Protected by L2</i>	LABEL
Rjaibi	55	L2
Bird	55	L2
Fielding	77	L3

Dropping protected data:

You cannot drop a column that is protected by a security label unless your LBAC credentials allow you to write to that column.

A column with a data type of DB2SECURITYLABEL cannot be dropped from a table. To remove it you must first drop the security policy from the table. When you drop the security policy the table is no longer protected with LBAC and the data type of the column is automatically changed from DB2SECURITYLABEL to VARCHAR(128) FOR BIT DATA. The column can then be dropped.

Your LBAC credentials do not prevent you from dropping entire tables or databases that contain protected data. If you would normally have permission to drop a table or a database you do not need any LBAC credentials to do so, even if the database contains protected data.

Related concepts:

- "Inserting of LBAC protected data" on page 563
- "Reading of LBAC protected data" on page 560
- "Updating of LBAC protected data" on page 565

Removal of LBAC protection from data

Removing a security policy from a table:

You must have SECADM authority to remove the security policy from a table. To remove the security policy from a table you use the DROP SECURITY POLICY clause of the ALTER TABLE statement. This also automatically removes protection from all rows and all columns of the table.

Removing protection from rows:

In a table that has protected rows every row must be protected by a security label. There is no way to remove LBAC protection from individual rows.

A column of type DB2SECURITYLABEL cannot be altered or removed except by removing the security policy from the table.

Removing protection from columns:

Protection of a column can be removed using the DROP COLUMN SECURITY clause of the SQL statement ALTER TABLE. To remove the protection from a column you must have LBAC credentials that allow you to read from and write to that column in addition to the normal privileges and authorities needed to alter a table.

Related concepts:

- “Label-based access control (LBAC) overview” on page 538
- “LBAC security labels” on page 547
- “LBAC security policies” on page 540
- “Protection of data using LBAC” on page 558

Related reference:

- “ALTER TABLE statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

Lightweight directory access protocol (LDAP) directory services

This section provides information to assist you in using Lightweight Directory Access Protocol (LDAP) directory services with the DB2 database management system.

Lightweight Directory Access Protocol (LDAP) overview

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:

- “Security considerations in an LDAP environment” on page 589
- “Extending the LDAP directory schema with DB2 object classes and attributes” on page 591
- “LDAP support and DB2 Connect” on page 589
- “Lightweight Directory Access Protocol (LDAP) directory service” on page 181
- “Security considerations for Active Directory” on page 590
- “Support for Active Directory” on page 575
- “DB2 registry and environment variables” in *Performance Guide*

Related tasks:

- “Attaching to a remote server in the LDAP environment” on page 583
- “Catalog a node alias for ATTACH” on page 581
- “Configuring DB2 in the IBM LDAP environment” on page 576
- “Configuring DB2 to use Active Directory” on page 576
- “Configuring the LDAP user for DB2 applications” on page 578
- “Creating an LDAP user” on page 577
- “Deregistering the database from the LDAP directory” on page 584
- “Deregistering the DB2 server” on page 582
- “Disabling LDAP support” on page 589
- “Enabling LDAP support after installation is complete” on page 588
- “Extending the directory schema for Active Directory” on page 591
- “Refreshing LDAP entries in local database and node directories” on page 584
- “Registering host databases in LDAP” on page 586
- “Registration of databases in the LDAP directory” on page 582
- “Registration of DB2 servers after installation” on page 578
- “Searching the LDAP servers” on page 585
- “Setting DB2 registry variables at the user level in the LDAP environment” on page 587
- “Update the protocol information for the DB2 server” on page 580

Related reference:

- “DB2 objects in the Active Directory” on page 593

- “LDAP object classes and attributes used by DB2” on page 598
- “Netscape LDAP directory support and attribute definitions” on page 593
- “Supported LDAP client and server configurations” on page 575

Supported LDAP client and server configurations

For the most up-to-date information about supported LDAP client and server configurations, see: <http://www.ibm.com/support/docview.wss?rs=71&uid=swg21233795>.

Note: When running on Windows operating systems, DB2 supports using either the IBM LDAP client or the Microsoft LDAP client. To explicitly select the IBM LDAP client, use the **db2set** command to set the DB2LDAP_CLIENT_PROVIDER registry variable to “IBM”. The Microsoft LDAP Client is included with the Windows operating system.

Related concepts:

- “Lightweight Directory Access Protocol (LDAP) overview” on page 573
- “Support for Active Directory” on page 575

Support for Active Directory

DB2 database system exploits the Active Directory as follows:

1. The DB2 database servers are published in the Active Directory as the `ibm_db2Node` objects. The `ibm_db2Node` object class is a subclass of the `ServiceConnectionPoint` (SCP) object class. Each `ibm_db2Node` object contains protocol configuration information to allow client applications to connect to the DB2 database server. When a new database is created, the database is published in the Active Directory as the `ibm_db2Database` object under the `ibm_db2Node` object.
2. When connecting to a remote database, DB2 client queries the Active Directory, through 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 `regsvr32` 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 database objects under the computer objects.

Note: If DB2 database is not installed on the domain controller, you can still view the property pages of DB2 database 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 590

Related tasks:

- “Configuring DB2 to use Active Directory” on page 576
- “Extending the directory schema for Active Directory” on page 591

Related reference:

- “DB2 objects in the Active Directory” on page 593

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 database 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.
3. Enable the LDAP support. For Windows 2000, Windows XP, or Windows Server 2003, the LDAP support is enabled by the installation program.
4. Log on to a domain user account when running DB2 database to read information from the Active Directory.

Related concepts:

- “DB2 registry and environment variables” in *Performance Guide*
- “Support for Active Directory” on page 575

Related tasks:

- “Configuring the LDAP user for DB2 applications” on page 578

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, LDAP support is enabled by the installation program. 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). You can 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 *Performance Guide*

Related tasks:

- "Configuring DB2 to use Active Directory" on page 576
- "Creating an LDAP user" on page 577

Related reference:

- "db2set - DB2 profile registry command" in *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 Linux and UNIX platforms.) User level support provides user-specific settings in a multi-user environment. An example is Windows 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 Tivoli® directory, you must define an LDAP user before you can store user-level information in LDAP. You can create an LDAP user by creating 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 Tivoli Directory Server is "LDIF2DB"

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 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 576
- “Configuring the LDAP user for DB2 applications” on page 578

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 the DB2 database manager, 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 Development,ou=Toronto,o=ibm,c=ca"
-w password
```

Related tasks:

- “Creating an LDAP user” on page 577

Related reference:

- “db2ldcfg - Configure LDAP environment command” in *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 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 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 TCP/IP"
```

The “<>” is substituted by the hostname on each machine where the **rah** command is run. In the rare occurrence where there are multiple DB2 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>
svcname <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.

To register the DB2 server in LDAP from a client application, call the `db2LdapRegister` API.

Related concepts:

- “rah and db2_all commands overview” on page 130

Related tasks:

- “Attaching to a remote server in the LDAP environment” on page 583
- “Catalog a node alias for ATTACH” on page 581
- “Deregistering the DB2 server” on page 582
- “Update the protocol information for the DB2 server” on page 580

Related reference:

- “CATALOG LDAP NODE command” in *Command Reference*
- “REGISTER command” in *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>
  svcname <tcpip_service_name>
```

Related tasks:

- “Attaching to a remote server in the LDAP environment” on page 583
- “Catalog a node alias for ATTACH” on page 581
- “Registration of DB2 servers after installation” on page 578

Related reference:

- “UPDATE LDAP NODE command” in *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 database in LDAP. Alternatively, you can call the db2LdapUpdateAlternateServerForDB API from a client application to update the alternate server for the database in LDAP.

Once established, this alternate server information is returned to the client upon connection.

Note:

It is strongly recommended to keep the alternate server information stored in the LDAP server synchronized with the alternate server information stored at the database server instance. Issuing the UPDATE ALTERNATE SERVER FOR DATABASE command (notice that it is not “FOR LDAP DATABASE”) at the database server instance will help ensure synchronization between the database server instance and the LDAP server.

When you enter UPDATE ALTERNATE SERVER FOR DATABASE command 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.

If the UPDATE ALTERNATE SERVER FOR LDAP DATABASE command is issued from an instance other than the database server instance, ensure the alternate server information is also identically configured at the database server instance using the UPDATE ALTERNATE SERVER FOR DATABASE command. After the client initially connects to the database server instance, the alternate server information returned from the database server instance will take precedence over what is configured in the LDAP server. If the database server instance has no alternate server information configured, client reroute will be considered disabled after the initial connect.

Related concepts:

- “Automatic client reroute description and setup” on page 45
- “Client reroute setup when using JCC Type 4 drivers” on page 54

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:

- “Attaching to a remote server in the LDAP environment” on page 583
- “Registration of DB2 servers after installation” on page 578

Related reference:

- “CATALOG LDAP NODE command” in *Command Reference*
- “UNCATALOG LDAP NODE command” in *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>
```

To deregister the DB2 server from LDAP from a client application, call the db2LdapDeregister API.

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 578

Related reference:

- “DEREGISTER command” in *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"
```

To register a database in LDAP from a client application, call the db2LdapCatalogDatabase API.

Related tasks:

- “Deregistering the database from the LDAP directory” on page 584
- “Registration of DB2 servers after installation” on page 578

Related reference:

- “CATALOG LDAP DATABASE command” in *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 mechanism 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 *Performance Guide*

Related tasks:

- “Catalog a node alias for ATTACH” on page 581
- “Registration of databases in the LDAP directory” on page 582
- “Registration of DB2 servers after installation” on page 578
- “Update the protocol information for the DB2 server” on page 580

Related reference:

- “ATTACH command” in *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>
```

To deregister a database from LDAP from a client application, call the `db2LdapUncatalogDatabase` API.

Related tasks:

- “Registration of databases in the LDAP directory” on page 582

Related reference:

- “UNCATALOG LDAP DATABASE command” in *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 mechanism 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 *Performance Guide*

Related reference:

- "dir_cache - Directory cache support configuration parameter" in *Performance Guide*
- "REFRESH LDAP command" in *Command Reference*

Searching the LDAP servers

Procedure:

The DB2 database system searches the current LDAP server (supported LDAP servers are: IBM Tivoli Directory Server, Microsoft Active Directory, and Sun One Directory Server) but in an environment where there are multiple LDAP servers, you can define the scope of the search. For example, if the information is not found in the current LDAP server, you can specify automatic search of all other LDAP servers, or, alternatively, you can restrict the search scope to only the current LDAP server, or to the local DB2 database catalog.

When you set the search scope, this sets the default search scope for the entire enterprise. The search scope is controlled through the DB2 database profile registry variable, DB2LDAP_SEARCH_SCOPE. To set the search scope value, 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”. If it is not set, the default value is “domain” which limits the search scope to the directory on the current LDAP server.

For example, you may want to initially set the search scope to “global” after a new database is created. This allows any DB2 client configured to use LDAP to search all the LDAP servers to find the database. Once the entry has been recorded on each machine after the first connect or attach for each client, if you have caching enabled, the search scope can be changed to “local”. Once changed to “local”, each client will not scan any LDAP servers.

Note: The DB2 database profile registry variables DB2LDAP_KEEP_CONNECTION and DB2LDAP_SEARCH_SCOPE are the only registry variables that can be set at the global level in LDAP.

Related concepts:

- “DB2 registry and environment variables” in *Performance Guide*

Related tasks:

- “Declaring, showing, changing, resetting, and deleting registry and environment variables” on page 68

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 through 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, the DB2 database system 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 partner1u 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 svcname 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 svcname 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:

- "CATALOG LDAP DATABASE command" in *Command Reference*
- "REGISTER command" in *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 systems.

DB2 has a caching mechanism. The DB2 profile registry variables at the user level are cached on the local machine. If the `-u1` 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, showing, changing, resetting, and deleting registry and environment variables” on page 68

Related reference:

- “db2set - DB2 profile registry command” in *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, 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). You can 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 *Performance Guide*

Related tasks:

- “Disabling LDAP support” on page 589

Related reference:

- “CATALOG LDAP DATABASE command” in *Command Reference*
- “db2set - DB2 profile registry command” in *Command Reference*
- “REGISTER command” in *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, showing, changing, resetting, and deleting registry and environment variables” on page 68
- “Enabling LDAP support after installation is complete” on page 588

Related reference:

- “DEREGISTER command” in *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:

- “Lightweight Directory Access Protocol (LDAP) overview” on page 573
- “Security considerations in an LDAP environment” on page 589

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 590

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 589

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 Tivoli Directory Server, 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 database attributes.

Related concepts:

- “Extending the directory schema for IBM Tivoli Directory Server” on page 595

Related tasks:

- “Extending the directory schema for Active Directory” on page 591

Extending the directory schema for Active Directory

Procedure:

Before DB2 database can store information in the Active Directory, the directory schema needs to be extended to include the new DB2 database object classes and attributes. The process of adding new object classes and attributes to the directory schema is called *schema extension*.

You must extend the schema for Active Directory by running the DB2 Schema Installation program, **db2schex** before the first installation of DB2 database 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 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 the DB2 database management system, 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 the DB2 database management system on Windows, when you run this same command that come with DB2 Version 8.2, it will add all the classes and attributes for DB2 database system LDAP support.

Examples:

- To install the DB2 database schema:

db2schex

- To install the DB2 database 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 database schema:

```
db2schex -u
```

- To uninstall the DB2 database schema and ignore errors:

```
db2schex -u -k
```

The DB2 Schema Installation program for Active Directory carries out the following tasks:

Notes:

1. Detects which server is the Schema Master
2. Binds to the Domain Controller that is the Schema Master
3. Ensures that the user has sufficient rights to add classes and attributes to the schema
4. Ensures that the schema master is writable (that is, the safety interlock in the registry is removed)
5. Creates all the new attributes
6. Creates all the new object classes
7. Detects errors, and if they occur, the program will roll back any changes to the schema.

Related concepts:

- “Extending the LDAP directory schema with DB2 object classes and attributes” on page 591

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 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 598

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 Sun One Directory Server 5.0, please refer to the topic about extending the directory schema for the Sun One Directory Server.

The DB2 attributes must be added to the `slapd.user_at.conf` as follows:

```
#####  
#  
# IBM DB2 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 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 concepts:

- “Extending the directory schema for Sun One Directory Server” on page 596

Related reference:

- “LDAP object classes and attributes used by DB2” on page 598

Extending the directory schema for IBM Tivoli Directory Server

If you are using IBM Tivoli Directory Server, all the object classes and attributes that are required by the DB2 database before Version 8.2 are included in the base schema. Run the following to extend the base schema with new DB2 database 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 directory schema for Sun One Directory Server” on page 596
- “Extending the LDAP directory schema with DB2 object classes and attributes” on page 591

Related tasks:

- “Extending the directory schema for Active Directory” on page 591

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 .\slldap-<machine_name>\config\schema.

On UNIX, if you have iPlanet installed in /usr/ipplanet/servers, add the above file to ./slapd-<machine_name>/config/schema.

The following is the contents of the file:

```
#####
# IBM DB2 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 60ibmdb2.1dif 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 directory schema for IBM Tivoli Directory Server” on page 595
- “Extending the LDAP directory schema with DB2 object classes and attributes” on page 591

Related tasks:

- “Extending the directory schema for Active Directory” on page 591

LDAP object classes and attributes used by DB2

The following tables describe the object classes that are used by the DB2 database manager:

Table 50. *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 51. *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 52. *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)	

Table 52. *eProperty* (continued)

Class	eProperty
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 53. *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
Special Notes	<ol style="list-style-type: none"> 1. The <i>DB2Node</i> class is derived from <i>eSap</i> object class under IBM Tivoli Directory Server and from <i>ServiceConnectionPoint</i> object class under Microsoft Active Directory. 2. The <i>host</i> is used under the IBM Tivoli Directory Server environment. The <i>dNSHostName</i> attribute is used under Microsoft Active Directory. 3. The <i>protocolInformation</i> is only used under the IBM Tivoli Directory Server environment. For Microsoft Active Directory, the attribute <i>ServiceBindingInformation</i>, inherited from the <i>ServiceConnectionPoint</i> class, is used to contain the protocol information.

The *protocolInformation* (in IBM Tivoli Directory Server) 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 client for Windows, if the APPN information is not configured on the local SNA stack; and, if the LAN adapter address and optional change password LU are found in LDAP, then the DB2 client tries to use this information to configure the SNA stack if it knows how to configure the stack.

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 54. 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

Table 54. DB2Database (continued)

Class	DB2Database
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 55. 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 56. db2authenticationLocation

Attribute	db2authenticationLocation
Active Directory LDAP Display Name	ibm-db2AuthenticationLocation
Active Directory Common Name (cn)	ibm-db2AuthenticationLocation
Description	Specifies where authentication takes place
Syntax	Case Ignore String
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 DCS ENCRYPT

Table 57. 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 58. 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 59. 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 60. 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

Table 60. *db2databaseRelease* (continued)

Attribute	db2databaseRelease
GUID (Global Unique Identifier)	b3afd31a-5c5b-11d3-b818-002035559151

Table 61. *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 62. *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 63. *db2nodePtr*

Attribute	db2nodePtr
Active Directory LDAP Display Name	ibm-db2NodePtr
Active Directory Common Name (cn)	ibm-db2NodePtr
Description	Pointer to the Node (DB2Node) object that represents the database server which owns the database
Syntax	Distinguished Name
Maximum Length	1000
Multi-Valued	Single-valued
OID (object identifier)	1.3.18.0.2.4.423
GUID (Global Unique Identifier)	b3afd31f-5c5b-11d3-b818-002035559151
Special Notes	This relationship allows the client to retrieve protocol communication information to connect to the database

Table 64. *db2altnodePtr*

Attribute	db2altnodePtr
Active Directory LDAP Display Name	ibm-db2AltNodePtr

Table 64. db2altnodePtr (continued)

Attribute	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 65. 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 66. 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 67. 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

Table 67. *db2instanceName* (continued)

Attribute	db2instanceName
OID (object identifier)	1.3.18.0.2.4.428
GUID (Global Unique Identifier)	b3afd31c-5c5b-11d3-b818-002035559151

Table 68. *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 69. *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 70. *cesProperty*

Attribute	cesProperty
Active Directory LDAP Display Name	ibm-cesProperty
Active Directory Common Name (cn)	ibm-cesProperty
Description	Values of this attribute may be used to provide application-specific preference configuration parameters. For example, a value may contain XML-formatted data. All values of this attribute must be homogeneous in the cesPropertyType attribute value.
Syntax	Case Exact String
Maximum Length	32700
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.307
GUID (Global Unique Identifier)	b3afd2d5-5c5b-11d3-b818-002035559151

Table 71. 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 72. 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 73. cisPropertyType

Attribute	cisPropertyType
Active Directory LDAP Display Name	ibm-cisPropertyType
Active Directory Common Name (cn)	ibm-cisPropertyType
Description	Values of this attribute may be used to describe the syntax, semantics, or other characteristics of all of the values of the cisProperty attribute. For example, a value of "INI File" might be used to indicate that all the values of the cisProperty attribute are INI files.
Syntax	Case Ignore String
Maximum Length	128
Multi-Valued	Multi-valued
OID (object identifier)	1.3.18.0.2.4.310
GUID (Global Unique Identifier)	b3afd2e1-5c5b-11d3-b818-002035559151

Table 74. 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 75. 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 76. 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 77. *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:

- “Lightweight Directory Access Protocol (LDAP) overview” on page 573

Tasks and required authorizations

Not all organizations divide job responsibilities in the same manner. Table 78 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 78. *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 SYSMANT authority over one or more databases. SYSCTRL authority in some cases.
System Operator	Monitors the database and carries out backup functions	SYSMANT 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.

Table 78. Common Job Titles, Tasks, and Required Authorization (continued)

JOB TITLE	TASKS	REQUIRED AUTHORIZATION
Information Center Consultant	Defines the data requirements for a query user; provides the data by creating tables and views and by granting access to database objects	DBADM authority over one or more databases.
Query User	Issues SQL statements to retrieve, add, delete, or change data; may save results as tables	CONNECT on one or more databases; CREATEIN on the schema of the tables and views being created; and, SELECT, INSERT, UPDATE, DELETE on some tables and views.

Note: If an application program contains dynamic SQL statements, the Program End User may need other privileges in addition to EXECUTE and CONNECT (such as SELECT, INSERT, DELETE, and UPDATE).

Related concepts:

- “Database administration authority (DBADM)” on page 509
- “Database authorities” on page 511
- “LOAD authority” on page 511
- “System administration authority (SYSADM)” on page 506
- “System control authority (SYSCTRL)” on page 507
- “System maintenance authority (SYSMAINT)” on page 508

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

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 created. This system catalog describes tables, columns, indexes, programs, privileges, and other objects.

The following views and table functions list information about privileges held by users, identities of users granting privileges, and object ownership:

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.PASSTHROUGHAUTH	Lists the server privilege
SYSCAT.ROUTINEAUTH	Lists the routine (functions, methods, and stored procedures) privileges

SYSCAT.SURROGATEAUTHIDS

Lists the authorization IDs for which another authorization ID can act as a surrogate.

Privileges granted to users by the system will have SYSIBM as the grantor. SYSADM, SYSMOINT SYSCTRL, and SYSMON 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 all names with DBADM authority” on page 611
- “Retrieving all privileges granted to users” on page 613
- “Retrieving authorization names with granted privileges” on page 610
- “Retrieving names authorized to access a table” on page 612
- “Securing the system catalog view” on page 613

Related reference:

- “SYSCAT.COLAUTH catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.DBAUTH catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.INDEXAUTH catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.PACKAGEAUTH catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.PASSTHROUGH AUTH catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.ROUTINEAUTH catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.SCHEMAAUTH catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.TABAUTH catalog view” in *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:

Starting with version 9.1 of the DB2 database manager, you can use the PRIVILEGES and other administrative views to retrieve information about the authorization names that have been granted privileges in a database. For example, the following query retrieves all explicit privileges and the authorization IDs to which they were granted, plus other information, from the PRIVILEGES administrative view:

```
SELECT AUTHID, PRIVILEGE, OBJECTNAME, OBJECTSCHEMA, OBJECTTYPE FROM SYSIBMADM.PRIVILEGES
```

The following query uses the AUTHORIZATIONIDS administrative view to find all the authorization IDs that have been granted privileges or authorities, and to show their types:

```
SELECT AUTHID, AUTHIDTYPE FROM SYSIBMADM.AUTHORIZATIONIDS
```


You can also use the SYSIBMADM.OBJECTOWNERS administrative view and the SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID table function to find security-related information.

Prior to version 9.1, no single system catalog view contained information about all privileges. For releases earlier than version 9.1, 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.PASSTHROUGHAUTH
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 609

Related reference:

- “AUTH_LIST_GROUPS_FOR_AUTHID table function – Retrieve group membership list for a given authorization ID” in *Administrative SQL Routines and Views*
- “AUTHORIZATIONIDS administrative view – Retrieve authorization IDs and types” in *Administrative SQL Routines and Views*
- “OBJECTOWNERS administrative view – Retrieve object ownership information” in *Administrative SQL Routines and Views*
- “PRIVILEGES administrative view – Retrieve privilege information” in *Administrative SQL Routines and Views*

Retrieving all names with DBADM authority

Procedure:

The following statement retrieves all authorization names that have been directly granted DBADM authority:

```
SELECT DISTINCT GRANTEE, GRANTEETYPE FROM SYSCAT.DBAUTH
WHERE DBADMAUTH = 'Y'
```

Note: This query will not return information about authorization names that acquired DBADM authority implicitly by having SYSADM authority.

Related concepts:

- “Database administration authority (DBADM)” on page 509
- “Using the system catalog for security issues” on page 609

Retrieving names authorized to access a table

Procedure:

Starting with version 9.1 of the DB2 database manager, you can use the PRIVILEGES and other administrative views to retrieve information about the authorization names that have been granted privileges in a database. The following statement retrieves all authorization names (and their types) that are directly authorized to access the table EMPLOYEE with the qualifier JAMES:

```
SELECT DISTINCT AUTHID, AUTHIDTYPE FROM SYSIBMADM.PRIVILEGES
WHERE OBJECTNAME = 'EMPLOYEE' AND OBJECTSCHEMA = 'JAMES'
```

For releases earlier than version 9.1, the following query retrieves the same information:

```
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 IN ('G','Y'))
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 515
- “Using the system catalog for security issues” on page 609

Related reference:

- “PRIVILEGES administrative view – Retrieve privilege information” in *Administrative SQL Routines and Views*

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. Starting with version 9.1 of the DB2 database manager, you can use the PRIVILEGES and other administrative views to retrieve information about the authorization names that have been granted privileges in a database. For example, the following query retrieves all the privileges granted to the current session authorization ID:

```
SELECT * FROM SYSIBMADM.PRIVILEGES
WHERE AUTHID = SESSION_USER AND AUTHIDTYPE = 'U'
```

The keyword SESSION_USER in this statement is a special register that is equal to the value of the current user's authorization name.

For releases earlier than version 9.1, the following examples provide similar information. For example, the following statement retrieves a list of the database privileges that have been directly granted to the individual authorization name JAMES:

```
SELECT * FROM SYSCAT.DBAUTH
WHERE GRANTEE = 'JAMES' AND GRANTEETYPE = 'U'
```

The following statement retrieves a list of the table privileges that were directly granted by the user JAMES:

```
SELECT * FROM SYSCAT.TBAUTH
WHERE GRANTOR = 'JAMES'
```

The following statement retrieves a list of the individual column privileges that were directly granted by the user JAMES:

```
SELECT * FROM SYSCAT.COLAUTH
WHERE GRANTOR = 'JAMES'
```

Related concepts:

- “Database authorities” on page 511
- “Authorization, privileges, and object ownership” on page 501
- “Using the system catalog for security issues” on page 609

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

Related reference:

- “PRIVILEGES administrative view – Retrieve privilege information” in *Administrative SQL Routines and Views*

Securing the system catalog view

As the system catalog views describe every object in the database, if you have sensitive data, you might want to restrict their access. Starting with version 9.1 of the DB2 database manager, you can use the CREATE DATABASE ... RESTRICTIVE command to create a database in which no privileges are automatically granted to PUBLIC. In this case, none of the following normal default grant actions occur:

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

If you have created a database using the RESTRICTIVE option, and you want to check that the permissions granted to PUBLIC are limited, you can issue the following query to verify which schemas PUBLIC can access:

```
SELECT DISTINCT OBJECTSCHEMA FROM SYSIBMADM.PRIVILEGES WHERE AUTHID='PUBLIC'
```

```
OBJECTSCHEMA
-----
SYSFUN
SYSIBM
SYSPROC
```

To see what access PUBLIC still has to SYSIBM, you can issue the following query to check what privileges are granted on SYSIBM. The results show that only EXECUTE on certain procedures and functions is granted.

```
SELECT * FROM SYSIBMADM.PRIVILEGES WHERE OBJECTSCHEMA = 'SYSIBM'
```

AUTHID	AUTHIDTYPE	PRIVILEGE	GRANTABLE	OBJECTNAME	OBJECTSCHEMA	OBJECTTYPE
PUBLIC	G	EXECUTE	N	SQL060207192129400	SYSPROC	FUNCTION
PUBLIC	G	EXECUTE	N	SQL060207192129700	SYSPROC	FUNCTION
PUBLIC	G	EXECUTE	N	SQL060207192129701	SYSPROC	
...						
PUBLIC	G	EXECUTE	Y	TABLES	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	TABLEPRIVILEGES	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	STATISTICS	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	SPECIALCOLUMNS	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	PROCEDURES	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	PROCEDURECOLS	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	PRIMARYKEYS	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	FOREIGNKEYS	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	COLUMNS	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	COLPRIVILEGES	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	UDTS	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	GETTYPEINFO	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	SQLCAMESSAGE	SYSIBM	PROCEDURE
PUBLIC	G	EXECUTE	Y	SQLCAMESSAGECCSID	SYSIBM	PROCEDURE

Note: The SYSIBMADM.PRIVILEGES administrative view is available starting with version 9.1 of the DB2 database manager.

Procedure for releases earlier than version 9.1:

For releases earlier than version 9.1 of the DB2 database manager, 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, if you don't want any user to be able to know what objects other users have access to, you should consider restricting access to the following catalog and administrative views:

- SYSCAT.COLAUTH
- SYSCAT.DBAUTH
- SYSCAT.INDEXAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.ROUTINEAUTH
- SYSCAT.SCHEMAAUTH
- SYSCAT.SECURITYLABELACCESS
- SYSCAT.SECURITYPOLICYEXEMPTIONS
- SYSCAT.SEQUENCEAUTH
- SYSCAT.SURROGATEAUTHIDS
- SYSCAT.TABAUTH
- SYSCAT.TBSPACEAUTH
- SYSCAT.XSROBJECTAUTH
- SYSIBMADM.AUTHORIZATIONIDS
- SYSIBMADM.OBJECTOWNERS
- SYSIBMADM.PRIVILEGES

This would prevent information on user privileges from becoming available to everyone with 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 equal to the value of the current session 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 view and base table by issuing the following two statements:

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC  
REVOKE SELECT ON TABLE SYSIBM.SYSTABAUTH FROM PUBLIC
```

Related concepts:

- “Catalog statistics” in *Performance Guide*
- “Database authorities” on page 511
- “Using the system catalog for security issues” on page 609

Related tasks:

- “Granting privileges” on page 519
- “Revoking privileges” on page 521

Related reference:

- “CREATE DATABASE command” in *Command Reference*
- “PRIVILEGES administrative view – Retrieve privilege information” in *Administrative SQL Routines and Views*

Security considerations

1. Gaining Access to Data through Indirect Means

The following are indirect means through which users can gain access to data they might not be authorized for:

- **Catalog views:** The DB2 database system catalog views store metadata and statistics about database objects. Users with SELECT access to the catalog views can gain some knowledge about data that they might not be qualified for. For better security, make sure that only qualified users have access to the catalog views.

Note: In DB2 Universal Database V8 or earlier, SELECT access on the catalog views was granted to PUBLIC by default. In DB2 V9.1 database systems, users can choose whether SELECT access to the catalog views is granted to PUBLIC or not by using the new RESTRICTIVE option on the CREATE DATABASE command.

- **Visual explain:** Visual explain shows the access plan chosen by the query optimizer for a particular query. The visual explain information also includes statistics about columns referenced in the query. These statistics can reveal information about a table's contents.
- **Explain snapshot:** The explain snapshot is compressed information that is collected when an SQL or XQuery statement is explained. It is stored as a binary large object (BLOB) in the EXPLAIN_STATEMENT table, and contains column statistics that can reveal information about table data. For better security, access to the explain tables should be granted to qualified users only.
- **Log reader functions:** A user authorized to run a function that reads the logs can gain access to data they might not be authorized for if they are able to

understand the format of a log record. These functions read the logs:

Function	Authority needed in order to execute the function
db2ReadLog	SYSADM or DBADM
db2ReadLogNoConn	None.

- **Replication:** When you replicate data, even the protected data is reproduced at the target location. For better security, make sure that the target location is at least as secure as the source location.
- **Exception tables:** When you specify an exception table while loading data into a table, users with access to the exception table can gain information that they might not be authorized for. For better security, only grant access to the exception table to authorized users and drop the exception table as soon as you are done with it.
- **Backup table space or database:** Users with the authority to run the backup command can take a backup of a database or a table space, including any protected data, and restore the data somewhere else. The backup can include data that the user might not otherwise have access to.

The backup command can be executed by users with SYSADM, SYSCTRL, or SYSMANT authority.

- **Set session authorization:** In DB2 Universal Database V8 or earlier a user with DBADM authority could use the SET SESSION AUTHORIZATION SQL statement to set the session authorization ID to any database user. In DB2 V9.1 database systems a user must be explicitly authorized through the GRANT SETSESSIONUSER statement before they can set the session authorization ID.

When migrating an existing database to a DB2 V9.1 database system, however, a user with existing explicit DBADM authority (for example, granted in SYSCAT.DBAUTH) will keep the ability to set the session authorization to any database user. This is allowed so that existing applications will continue to work. Being able to set the session authorization potentially allows access to all protected data. For more restrictive security, you can override this setting by executing the REVOKE SETSESSIONUSER SQL statement.

- **Statement and deadlock monitoring:** As part of the deadlock monitoring activity of DB2 database management systems, values associated with parameter markers are written to the monitoring output when the WITH VALUES clause is specified. A user with access to the monitoring output can gain access to information for which they might not be authorized.
- **Traces:** A trace can contain table data. A user with access to such a trace can gain access to information that they might not be authorized for.
- **Dump files:** To help in debugging certain problems, DB2 database products might generate memory dump files in the sqllib\db2dump directory. These memory dump files might contain table data. If they do, users with access to the files can gain access to information that they might not be authorized for. For better security you should limit access to the sqllib\db2dump directory.
- **db2dart:** The db2dart tool examines a database and reports any architectural errors that it finds. The tool can access table data and DB2 does not enforce access control for that access. A user with the authority to run the db2dart tool or with access to the db2dart output can gain access to information that they might not be authorized for.

- **REOPT bind option:** When the REOPT bind option is specified, explain snapshot information for each reoptimizable incremental bind SQL statement is placed in the explain tables at run time. The explain will also show input data values.
 - **db2cat:** The db2cat tool is used to dump a table's packed descriptor. The table's packed descriptor contains statistics that can reveal information about a table's contents. A user who runs the db2cat tool or has access to the output can gain access to information that they might not be authorized for.
2. Default Privileges Granted Upon Creating a Database

The following are the default privileges to certain system tables granted when a database is created:

1. SYSIBM.SYSDBAUTH

- The database creator is granted the following privileges:
 - DBADM
 - CREATETAB
 - CREATEROLE
 - BINDADD
 - CONNECT
 - NOFENCE
 - IMPLSCHEMA
 - LOAD
 - EXTERNALROUTINE
 - QUIESCECONNECT
- The special group PUBLIC is granted the following privileges:
 - CREATETAB
 - BINDADD
 - CONNECT
 - IMPLSCHEMA

2. SYSIBM.SYSTABAUTH

- The special group PUBLIC is granted the following privileges:
 - SELECT on all SYSCAT and SYSIBM tables
 - SELECT and UPDATE on all SYSSTAT tables

3. SYSIBM.SYSROUTINEAUTH

- The special group PUBLIC is granted the following privileges:
 - EXECUTE with GRANT on all procedures in schema
 - SQLJ EXECUTE with GRANT on all functions and procedures in schema SYSFUN
 - EXECUTE with GRANT on all functions and procedures in schema SYSPROC
 - EXECUTE on all table functions in schema SYSIBM
 - EXECUTE on all other procedures in schema SYSIBM

4. SYSIBM.SYSPACKAGEAUTH

- The database creator is granted the following privileges:
 - CONTROL on all packages created in the NULLID schema
 - BIND with GRANT on all packages created in the NULLID schema
 - EXECUTE with GRANT on all packages created in the NULLID schema

-
- The special group PUBLIC is granted the following privileges:
 - BIND on all packages created in the NULLID schema
 - EXECUTE on all packages created in the NULLID schema
- 5. SYSIBM.SCHEMAAUTH
 - The special group PUBLIC is granted the following privileges:
 - CREATEIN on schema SQLJ
 - CREATE IN on schema NULLID
- 6. SYSIBM.TBSPACEAUTH
 - The special group PUBLIC is granted the following privileges:
 - USE on table space USERSPACE1

Related concepts:

- “Explain snapshot” on page 456
- “Visual Explain” on page 463

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:

- “Application proxy firewalls” on page 620
- “Circuit level firewalls” on page 620
- “Screening router firewalls” on page 619
- “Stateful multi-layer inspection (SMLI) firewalls” on page 620

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 database are open for incoming and outgoing packets. DB2 database uses port 523 for the DB2 Administration Server (DAS), which is used by the DB2 database 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 619
-

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 database 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 619
-

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 database supports SOCKS Version 4.

Related concepts:

- “Introduction to firewall support” on page 619
-

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 619

Chapter 9. Auditing DB2 database activities

DB2 database auditing activities are shown in this section.

Introduction to the DB2 database 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 database 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 database 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 database partition at which the user is connected (the coordinator partition) or at the catalog partition (if they are not the same database partition). The implication of this is that audit records can be generated by more than one database partition. Part of each audit record contains information on the coordinator partition and originating database partition 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 database 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 database instance.
- Stop recording auditable events within the DB2 database 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 database 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, a query statement for dynamic queries, a package identifier for static queries, or an indicator of the type of operation being performed, such as `CONNECT`, can provide needed context when analyzing audit results.

Note: The SQL or XQuery 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 623

- “Audit facility record layouts (introduction)” on page 636
- “Audit facility tips and techniques” on page 654

Related tasks:

- “Controlling DB2 database audit facility activities” on page 655

Related reference:

- “Audit facility messages” on page 636
- “Audit facility usage” on page 624

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 database that can operate even if the DB2 database 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 database 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 database 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 database. 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 query statement and the audit settings for the DB2 database 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 or XQuery 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 or XQuery statements in the package is carried out at the time the package is precompiled or bound. The execution of the static SQL or XQuery 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 or XQuery 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:

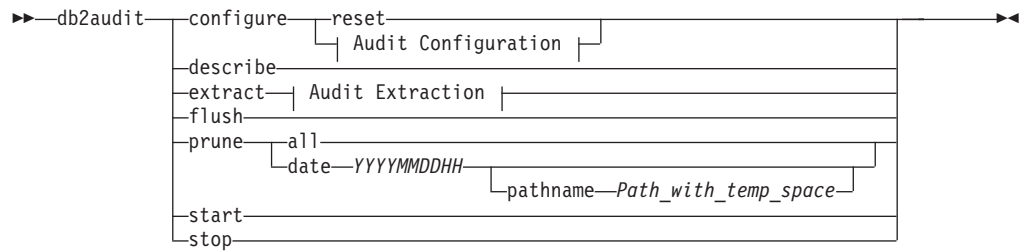
- "Introduction to the DB2 database audit facility" on page 621

Related reference:

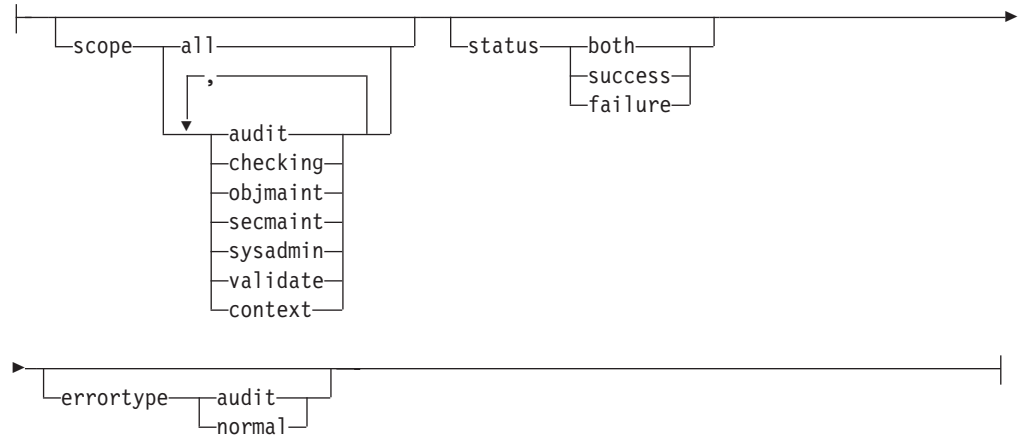
- "audit_buf_sz - Audit buffer size configuration parameter" in *Performance Guide*
- "Audit facility usage" on page 624

Audit facility usage

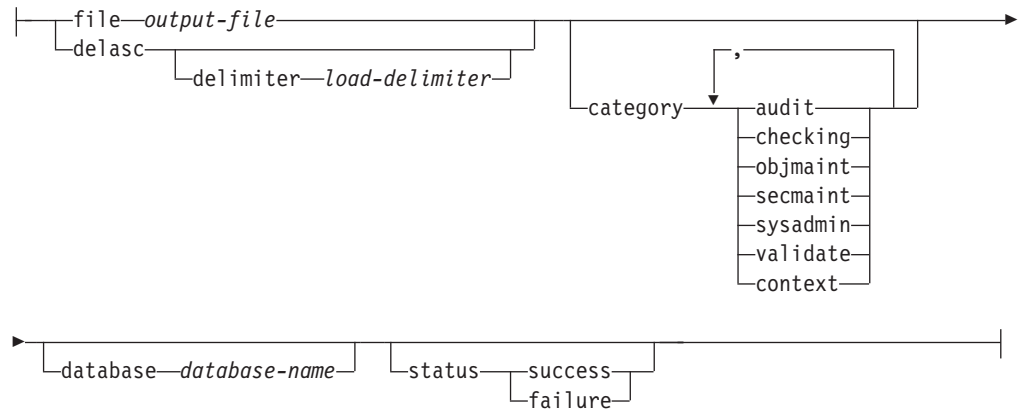
A review of each part of the following syntax diagrams will assist you in the understanding of how the audit facility can be used.



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 database across all database 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 database 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 sql1ib. 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 database 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 database instance, auditing will begin on all database 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 database instance, auditing will be stopped on all database 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:

- “Audit facility tips and techniques” on page 654
- “Introduction to the DB2 database audit facility” on page 621

Related reference:

- “db2audit - Audit facility administrator tool command” in *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 Select the table data.

Related concepts:

- “Audit facility behavior” on page 623
- “Audit facility tips and techniques” on page 654

Related tasks:

- “Creating tables to hold the DB2 audit data” on page 628
- “Creating DB2 audit data files” on page 631
- “Loading DB2 audit data into tables” on page 632
- “Selecting DB2 audit data from tables” on page 635

Related reference:

- “Audit facility usage” on page 624

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 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),  
                      PKGSHEMA 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),  
                      CHKAUTHID VARCHAR(128))
```

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),
PKGSHEMA VARCHAR(128),
PKGNAME VARCHAR(128),
PKGSECNUM SMALLINT,
OBJSHEMA 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),
PKGSHEMA VARCHAR(128),
PKGNAME VARCHAR(128),
PKGSECNUM SMALLINT,
OBJSHEMA 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),
PKGSHEMA 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),
PKGSHEMA VARCHAR(128),
PKGNAME VARCHAR(128),
PKGSECNUM SMALLINT,
PKGVER VARCHAR(64),
PLUGINNAME VARCHAR(32))
```

- 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),
PKGSHEMA VARCHAR(128),
PKGNAME VARCHAR(128),
PKGSECNUM SMALLINT,
STMTTEXT CLOB(2M),
PKGVER VARCHAR(64))
```

- After creating the tables, issue the COMMIT statement to ensure that the table definitions are written to disk.
- 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:

- “Creating DB2 audit data files” on page 631
- “Setting a schema” on page 169

Related reference:

- “CREATE SCHEMA statement” in *SQL Reference, Volume 2*
- “CREATE TABLE statement” in *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:

- 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 `sql1ib`. 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 tasks:

- “Loading DB2 audit data into tables” on page 632

Related reference:

- “Audit record layout for SYSADMIN events” on page 650
- “Audit record layout for VALIDATE events” on page 651
- “db2audit - Audit facility administrator tool command” in *Command Reference*
- “Audit facility usage” on page 624
- “Audit record layout for AUDIT events” on page 637
- “Audit record layout for CHECKING events” on page 638
- “Audit record layout for CONTEXT events” on page 652
- “Audit record layout for OBJMAINT events” on page 643
- “Audit record layout for SECMAINT events” on page 645

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, use the fully qualified path name. For example, if you have DB2 database 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 tablesselect 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 concepts:

- “Privileges, authorities, and authorizations required to use Load” in *Data Movement Utilities Guide and Reference*
- “Load considerations for MDC tables” in *Administration Guide: Planning*
- “LOAD authority” on page 511

Related tasks:

- “Selecting DB2 audit data from tables” on page 635
- “Enabling parallelism for loading data” on page 10
- “Loading data into a table using the Load wizard” on page 237

Related reference:

- “File type modifiers for the load utility” in *Command Reference*

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 SQL 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 *SQL Reference, Volume 1*
- “SELECT statement” in *SQL Reference, Volume 2*
- “Audit record layout for AUDIT events” on page 637
- “Audit record layout for CHECKING events” on page 638
- “Audit record layout for CONTEXT events” on page 652
- “Audit record layout for OBJMAINT events” on page 643
- “Audit record layout for SECMAINT events” on page 645
- “Audit record layout for SYSADMIN events” on page 650
- “Audit record layout for VALIDATE events” on page 651
- “List of possible CHECKING access approval reasons” on page 640
- “List of possible CHECKING access attempted types” on page 641
- “List of possible CONTEXT audit events” on page 653
- “List of possible SECMAINT privileges or authorities” on page 647
- “List of possible SYSADMIN audit events” on page 650

Audit facility messages

SQL1322N An error occurred when writing to the audit log file.

Explanation: The DB2 database 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

Related concepts:

- “Introduction to the DB2 database audit facility” on page 621

SQL1323N An error occurred when accessing the audit configuration file.

Explanation: The audit configuration file (db2audit.cfg) could not be opened, or was invalid. Possible reasons for this error are that the db2audit.cfg file either does not exist, or has been damaged.

User response: Take one of the following actions:

- Restore from a saved version of the file.
- Reset the audit facility configuration file by issuing
db2audit reset

sqlcode: -1323

sqlstate: 57019

Audit facility record layouts (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 637
- “Audit record layout for CHECKING events” on page 638
- “Audit record layout for CONTEXT events” on page 652
- “Audit record layout for OBJMAINT events” on page 643
- “Audit record layout for SECMAINT events” on page 645
- “Audit record layout for SYSADMIN events” on page 650
- “Audit record layout for VALIDATE events” on page 651

Details on audit facility record layouts

The various audit facility record layouts are shown in this section.

Audit record layout for AUDIT events

Sample audit record:

```
timestamp=1998-06-24-11.54.05.151232;category=AUDIT;audit event=START;
event correlator=0;event status=0;
userid=boss;authid=BOSS;
```

Table 79. Audit Record Layout for AUDIT Events

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 636

Audit record layout for CHECKING events

Sample audit record:

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
event correlator=2;event status=0;
database=F00;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;
```

Table 80. Audit record layout for CHECKING events

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, CHECKING_FUNCTION, and CHECKING_TRANSFER
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 80. Audit record layout for CHECKING events (continued)

NAME	FORMAT	DESCRIPTION
Package Version	VARCHAR (64)	Version of the package in use at the time that the audit event occurred.
Checked Authorization ID	VARCHAR(128)	Authorization ID is checked when it is different than the authorization ID at the time of the audit event. For example, this can be the target owner in a TRANSFER OWNERSHIP statement.

Related concepts:

- “Audit facility record layouts (introduction)” on page 636

Related reference:

- “Audit record object types” on page 639
- “List of possible CHECKING access approval reasons” on page 640
- “List of possible CHECKING access attempted types” on page 641

Audit record object types

Table 81. 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	
UNIQUE_CONSTRAINT		X	

Table 81. Audit Record Object Types Based on Audit Events (continued)

Object type	CHECKING events	OBJMAINT events	SECMAINT events
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		
SECURITY_LABEL		X	X
SECURITY_POLICY		X	X
SECURITY_LABEL_COMPONENT		X	
ACCESS_RULE			X
XSR object	X	X	X

Related reference:

- “Audit record layout for CHECKING events” on page 638
- “Audit record layout for OBJMAINT events” on page 643
- “Audit record layout for SECMAINT events” on page 645

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 or user has SYSADM authority.

0x0000000000000004 SYCTRL

Access is approved; the application or user has SYCTRL authority.

0x0000000000000008 SYSMANT

Access is approved; the application or user has SYSMANT authority.

0x0000000000000010 DBADM

Access is approved; the application or user has DBADM authority.

0x0000000000000020 DATABASE PRIVILEGE

Access is approved; the application or user has an explicit privilege on the database.

0x0000000000000040 OBJECT PRIVILEGE

Access is approved; the application or user has an explicit privilege on the object or function.

0x0000000000000080 DEFINER

Access is approved; the application or user is the definer of the object or function.

0x0000000000000100 OWNER

Access is approved; the application or user is the owner of the object or function.

0x0000000000000200 CONTROL

Access is approved; the application or user has CONTROL privilege on the object or function.

0x0000000000000400 BIND

Access is approved; the application or user has bind privilege on the package.

0x0000000000000800 SYSQUIESCE

Access is approved; if the instance or database is in quiesce mode, the application or user may connect or attach.

0x0000000000001000 SYSMON

Access is approved; the application or user has SYSMON authority.

0x0000000000002000 SECADM

Access is approved; the application or user has SECADM authority.

0x0000000000004000 SETSESSIONUSER

Access is approved; the application or user has SETSESSIONUSER authority.

Related reference:

- “Audit record layout for CHECKING events” on page 638
- “List of possible CHECKING access attempted types” on page 641

List of possible CHECKING access attempted types

The following is the list of possible CHECKING access attempted types. If Audit Event is CHECKING_TRANSFER, then the audit entry reflects that a privilege is held or not.

0x0000000000000001 CONTROL

Attempt to verify if CONTROL privilege is held.

0x0000000000000002 ALTER

Attempt to alter an object or to verify if ALTER privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000004 DELETE

Attempt to delete an object or to verify if DELETE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000008 INDEX

Attempt to use an index or to verify if INDEX privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000010 INSERT

Attempt to insert into an object or to verify if INSERT privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000020 SELECT
 Attempt to query a table or view or to verify if SELECT privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000040 UPDATE
 Attempt to update data in an object or to verify if UPDATE privilege is held if Audit Event is CHECKING_TRANSFER.

0x0000000000000080 REFERENCE
 Attempt to establish referential constraints between objects or to verify if REFERENCE privilege is held if Audit Event is CHECKING_TRANSFER.

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 or to verify if EXECUTE privilege is held if Audit Event is CHECKING_TRANSFER.

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.

0x000000001000000 Access All
 Attempt to execute a statement with all required privileges on objects held (only used for DBADM/SYSADM).

0x000000002000000 Drop All
 Attempt to drop multiple objects.

0x000000004000000 LOAD
 Attempt to load a table in a table space.

0x000000008000000 USE
 Attempt to create a table in a table space or to verify if USE privilege is held if Audit Event is CHECKING_TRANSFER.

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.

0x00000000400000000 TRANSFER
 Attempt to transfer an object.

0x00000000800000000 ALTER_WITH_GRANT
 Attempt to verify if ALTER with GRANT privilege is held.

0x00000001000000000 DELETE_WITH_GRANT
 Attempt to verify if DELETE with GRANT privilege is held.

0x00000002000000000 INDEX_WITH_GRANT
 Attempt to verify if INDEX with GRANT privilege is held

0x00000004000000000 INSERT_WITH_GRANT
 Attempt to verify if INSERT with GRANT privilege is held.

0x00000008000000000 SELECT_WITH_GRANT
 Attempt to verify if SELECT with GRANT privilege is held.

0x00000010000000000 UPDATE_WITH_GRANT
 Attempt to verify if UPDATE with GRANT privilege is held.

0x00000020000000000 REFERENCE_WITH_GRANT
 Attempt to verify if REFERENCE with GRANT privilege is held.

0x00000040000000000 USAGE
 Attempt to use a sequence or an XSR object or to verify if USAGE privilege is held if Audit Event is CHECKING_TRANSFER.

Related reference:

- “Audit record layout for CHECKING events” on page 638
- “List of possible CHECKING access approval reasons” on page 640

Audit record layout for OBJMAINT events

Sample audit record:

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
event correlator=3;event status=0;
database=F00;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;

```

Table 82. Audit Record Layout for OBJMAINT Events

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, DROP_OBJECT, and ALTER_OBJECT. ALTER_OBJECT events are generated only when altering protected tables.
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.
Security Policy Name	VARCHAR(128)	The name of the security policy if the object type is TABLE and that table is associated with a security policy.

Table 82. Audit Record Layout for OBJMAINT Events (continued)

NAME	FORMAT	DESCRIPTION
Alter Action	VARCHAR(32)	Specific Alter operation Possible values include: <ul style="list-style-type: none"> • ADD_PROTECTED_COLUMN • ADD_COLUMN_PROTECTION • DROP_COLUMN_PROTECTION • ADD_ROW_PROTECTION • ADD_SECURITY_POLICY
Protected Column Name	VARCHAR(128)	If the Alter Action is ADD_COLUMN_PROTECTION or DROP_COLUMN_PROTECTION this is the name of the affected column.
Column Security Label	VARCHAR(128)	The security label protecting the column specified in the field Column Name.
Security Label Column Name	VARCHAR(128)	Name of the column containing the security label protecting the row.

Related concepts:

- “Introduction to the DB2 database audit facility” on page 621

Related reference:

- “Audit record object types” on page 639

Audit record layout for SECMAINT events

Sample audit record:

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
event correlator=4;event status=0;
database=F00;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;
```

Table 83. Audit Record Layout for SECMAINT Events

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, UPDATE_DBM_CFG, and TRANSFER_OWNERSHIP.
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

Table 83. Audit Record Layout for SECMAINT Events (continued)

NAME	FORMAT	DESCRIPTION
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. If the object type field is ACCESS_RULE then this field contains the security policy name associated with the rule. The name of the rule is stored in the field Object Name. If the object type field is SECURITY_LABEL, then this field contains the name of the security policy that the security label is part of. The name of the security label is stored in the field Object Name.
Object Name	VARCHAR (128)	Name of object for which the audit event was generated. If the object type field is ACCESS_RULE then this field contains the name of the rule. The security policy name associated with the rule is stored in the field Object Schema. If the object type field is SECURITY_LABEL, then this field contains the name of the security label. The name of the security policy that it is part of is stored in the field Object Schema.
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.
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.
Access Type	VARCHAR(32)	The access type for which a security label is granted. Possible values: <ul style="list-style-type: none"> • READ • WRITE • ALL

Table 83. Audit Record Layout for SECMAINT Events (continued)

NAME	FORMAT	DESCRIPTION
Assumable Authid	VARCHAR(128)	When the privilege granted is a SETSESSIONUSER privilege this is the authorization ID that the grantee is allowed to set as the session user.

Related concepts:

- “Audit facility record layouts (introduction)” on page 636

Related reference:

- “Audit record object types” on page 639
- “List of possible SECMAINT privileges or authorities” on page 647

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.

0x0000008000000000 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.

0x0000001000000000 Package BIND with GRANT
BIND privilege granted or revoked on a package with granting of privileges allowed.

0x0000002000000000 EXECUTE
EXECUTE privilege granted or revoked on a package or a routine.

0x0000004000000000 EXECUTE with GRANT
EXECUTE privilege granted or revoked on a package or a routine with granting of privileges allowed.

0x0000008000000000 EXECUTE IN SCHEMA
EXECUTE privilege granted or revoked for all routines in a schema.

0x0000001000000000 EXECUTE IN SCHEMA with GRANT
EXECUTE privilege granted or revoked for all routines in a schema with granting of privileges allowed.

0x0000002000000000 EXECUTE IN TYPE
EXECUTE privilege granted or revoked for all routines in a type.

0x0000004000000000 EXECUTE IN TYPE with GRANT
EXECUTE privilege granted or revoked for all routines in a type with granting of privileges allowed.

0x0000008000000000 CREATE EXTERNAL ROUTINE
CREATE EXTERNAL ROUTINE privilege granted or revoked.

0x0000001000000000 QUIESCE_CONNECT
QUIESCE_CONNECT privilege granted or revoked.

0x0000004000000000 SECADM Authority
SECADM authority granted or revoked

0x0000004000000000 SETSESSIONUSER Privilege
SETSESSIONUSER granted or revoked

0x0000008000000000 Exemption
Exemption granted or revoked

0x0100000000000000 Security label
Security label granted or revoked

Related reference:

- “Audit record layout for SECMAINT events” on page 645

Audit record layout for SYSADMIN events

Sample audit record:

```
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;
```

Table 84. Audit Record Layout for SYSADMIN Events

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 636

Related reference:

- “List of possible SYSADMIN audit events” on page 650

List of possible SYSADMIN audit events

The following is the list of possible SYSADMIN audit events:

Table 85. SYSADMIN Audit Events

START_DB2	ROLLFORWARD_DB
STOP_DB2	SET_RUNTIME_DEGREE
CREATE_DATABASE	SET_TABLESPACE_CONTAINERS
ALTER_DATABASE	UNCATALOG_DB
DROP_DATABASE	UNCATALOG_DCS_DB
UPDATE_DBM_CFG	UNCATALOG_NODE
UPDATE_DB_CFG	UPDATE_ADMIN_CFG
CREATE_TABLESPACE	UPDATE_MON_SWITCHES
DROP_TABLESPACE	LOAD_TABLE
ALTER_TABLESPACE	DB2AUDIT
RENAME_TABLESPACE	SET_APPL_PRIORITY
CREATE_NODEGROUP	CREATE_DB_AT_NODE
DROP_NODEGROUP	KILLDBM
ALTER_NODEGROUP	MIGRATE_SYSTEM_DIRECTORY
CREATE_BUFFERPOOL	DB2REMOT
DROP_BUFFERPOOL	DB2AUD
ALTER_BUFFERPOOL	MERGE_DBM_CONFIG_FILE
CREATE_EVENT_MONITOR	UPDATE_CLI_CONFIGURATION
DROP_EVENT_MONITOR	OPEN_TABLESPACE_QUERY
ENABLE_MULTIPAGE	SINGLE_TABLESPACE_QUERY
MIGRATE_DB_DIR	CLOSE_TABLESPACE_QUERY
DB2TRC	FETCH_TABLESPACE
DB2SET	OPEN_CONTAINER_QUERY
ACTIVATE_DB	FETCH_CONTAINER_QUERY
ADD_NODE	CLOSE_CONTAINER_QUERY
BACKUP_DB	GET_TABLESPACE_STATISTICS
CATALOG_NODE	DESCRIBE_DATABASE
CATALOG_DB	ESTIMATE_SNAPSHOT_SIZE
CATALOG_DCS_DB	READ_ASYNC_LOG_RECORD
CHANGE_DB_COMMENT	PRUNE_RECOVERY_HISTORY
DEACTIVATE_DB	UPDATE_RECOVERY_HISTORY
DROP_NODE_VERIFY	QUIESCE_TABLESPACE
FORCE_APPLICATION	UNLOAD_TABLE
GET_SNAPSHOT	UPDATE_DATABASE_VERSION
LIST_DRDA_INDOUBT_TRANSACTIONS	CREATE_INSTANCE
MIGRATE_DB	DELETE_INSTANCE
RESET_ADMIN_CFG	SET_EVENT_MONITOR
RESET_DB_CFG	GRANT_DBADM
RESET_DBM_CFG	REVOKE_DBADM
RESET_MONITOR	GRANT_DB_AUTHORITIES
RESTORE_DB	REVOKE_DB_AUTHORITIES
	REDIST_NODEGROUP

Related reference:

- “Audit record layout for SYSADMIN events” on page 650

Audit record layout for VALIDATE events

Sample audit record:

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;audit event=CHECK_GROUP_MEMBERSHIP;
event correlator=2;event status=-1092;
database=F00;userid=boss;authid=BOSS;execution id=newton;
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;
```

Table 86. Audit Record Layout for VALIDATE Events

NAME	FORMAT	DESCRIPTION
Timestamp	CHAR(26)	Date and time of the audit event.
Category	CHAR(8)	Category of audit event. Possible values are: VALIDATE
Audit Event	VARCHAR(32)	Specific Audit Event. Possible values include: GET_GROUPS, GET_USERID, AUTHENTICATE_PASSWORD, VALIDATE_USER, and GET_USERMAPPING_FROM_PLUGIN.
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 636

Audit record layout for CONTEXT events

Sample audit record:

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;audit event=EXECUTE_IMMEDIATE;
event correlator=3;
database=F00;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);
```

Table 87. Audit Record Layout for CONTEXT Events

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 or XQuery statement, if applicable. Null if no SQL or XQuery 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 636

Related reference:

- “List of possible CONTEXT audit events” on page 653

List of possible CONTEXT audit events

The following is the list of possible CONTEXT audit events:

Table 88. 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 652

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.

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 database 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 636

Related reference:

- "Audit facility usage" on page 624

Controlling DB2 database 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=F00;
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=F00;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=F00;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=F00;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=F00;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=F00;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=F00;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=F00;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=F00;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=F00;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=F00;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=F00;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 636

Related reference:

- “Audit facility usage” on page 624

Part 3. Appendixes

Appendix A. Conforming to the naming rules

General naming rules

Rules exist for the naming of all objects, users and groups. 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 lowercase letters in a name.

- On UNIX platforms, names must be in lowercase.
- 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 the DB2 database. 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.

User and group names also need to follow the rules forced on specific operation systems by the related systems. For example, on Linux and UNIX platforms, user names and primary group names must follow these rules:

- Allowed characters: lowercase a through z, 0 through 9, and _ (underscore) for names not starting with 0 through 9.
- Length must be less than or equal to 8 characters.

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 database object naming rules” on page 663
- “Federated database object naming rules” on page 666
- “User, user ID and group naming rules” on page 666
- “Workstation naming rules” on page 667

DB2 database object naming rules

All objects follow the General Naming Rules. In addition, some objects have additional restrictions shown in the accompanying tables.

Table 89. Database, database alias and instance naming rules

Objects	Guidelines
<ul style="list-style-type: none"> • Databases • Database aliases • Instances 	<ul style="list-style-type: none"> • Database names must be unique within the location in which they are cataloged. On Linux and UNIX implementations of the DB2 database manager, 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, no instance can have the same name as a service name. <p>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.</p>

Table 90. Database object naming rules

Objects	Guidelines
<ul style="list-style-type: none"> • Aliases • Buffer pools • Columns • Event monitors • Indexes • Methods • Nodegroups • Packages • Package versions • Schemas • Stored procedures • Tables • Table spaces • Triggers • UDFs • UDTs • Views 	<p>Can contain up to 18 bytes <i>except</i> for the following:</p> <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> – 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 91. Federated database object naming rules

Objects	Guidelines
<ul style="list-style-type: none"> • Function mappings • Index specifications • Nicknames • Servers • Type mappings • User mappings • Wrappers 	<ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> – 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 663

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 663

User, user ID and group naming rules

Table 92. User, user ID and group naming rules

Objects	Guidelines
<ul style="list-style-type: none">• Group names• User names• User IDs	<ul style="list-style-type: none">• Group names can contain up to 30 characters.• User IDs on Linux and UNIX operating systems can contain up to 8 characters.• User names on Windows can contain up to 30 characters.• When not using Client authentication, non-Windows 32-bit clients connecting to Windows with user names longer than 8 characters are supported when the user name and password are specified explicitly.• Names and IDs cannot:<ul style="list-style-type: none">– Be USERS, ADMINS, GUESTS, PUBLIC, LOCAL or any SQL reserved word– Begin with IBM, SQL or SYS.

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:

- “Federated database object naming rules” on page 666
- “General naming rules” on page 663

Federated database object naming rules

Table 93. Federated database object naming rules

Objects	Guidelines
<ul style="list-style-type: none">• Function mappings• Index specifications• Nicknames• Servers• Type mappings• User mappings• Wrappers	<ul style="list-style-type: none">• 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:<ul style="list-style-type: none">– Valid accented letters (such as ö)– Multibyte characters, except multibyte spaces (for multibyte environments)

Related concepts:

- “General naming rules” on page 663

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 663

Maintaining passwords on servers

You might 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 database system 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 Configuration Assistant (CA) can also be used to change the password.

Related concepts:

- “Additional restrictions and recommendations regarding the use of schema names” on page 667
- “DB2 database object naming rules” on page 663
- “Delimited identifiers and object names” on page 665
- “Federated database object naming rules” on page 666
- “General naming rules” on page 663
- “User, user ID and group naming rules” on page 666
- “Workstation naming rules” on page 667

Workstation naming rules

A *workstation name* specifies the NetBIOS name for a database server, database client, or DB2 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 <i>nname</i>	Node Number	Derived Node NetBIOS <i>nname</i>
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 663

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:

- “DB2 database object naming rules” on page 663
- “General naming rules” on page 663
- “Workstation naming rules” on page 667

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 the DB2 database system.

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:

- “DB2 database object naming rules” on page 663
- “General naming rules” on page 663
- “Workstation naming rules” on page 667

Appendix B. 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 database system integration with Windows Management Instrumentation” on page 672

Related reference:

- “Windows Management Instrumentation samples” in *Samples Topics*

DB2 database system 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 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 Event Log provider.

DB2 database system 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 distributed. 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 671

Related reference:

- “Windows Management Instrumentation samples” in *Samples Topics*

Appendix C. Using Windows security

DB2 and Windows security introduction

A Windows 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.

Note: Two-part user IDs are supported by the CONNECT statement and the ATTACH command. The qualifier of the SAM-compatible user ID is the NetBIOS style name which has a maximum length of 15 characters.

During the setup procedure when a Windows 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 Domain, the user may log on to the Domain. 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:

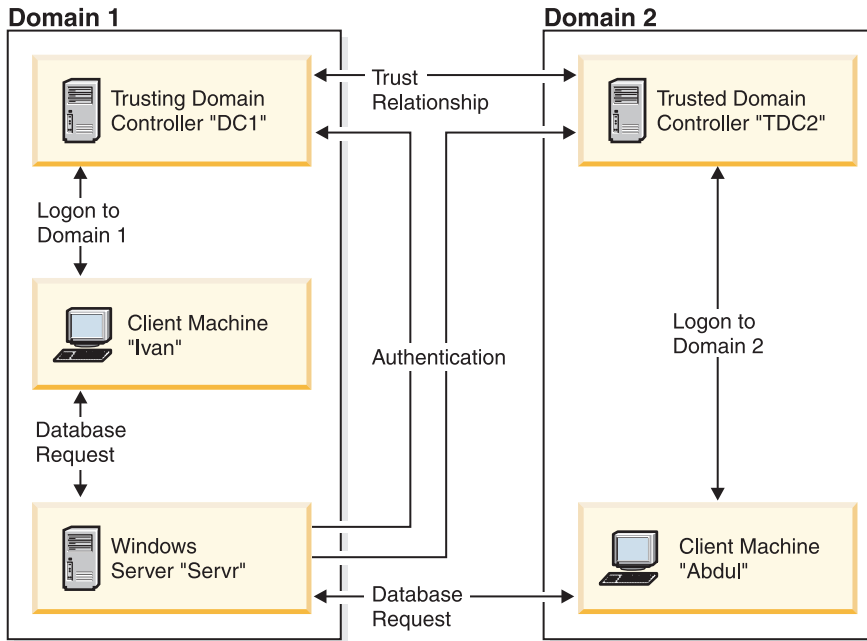


Figure 7. Authentication Using Windows Domains

Each machine has a security database, Security Access Management (SAM). DC1 is the domain controller, in which the client machine, Ivan, and the DB2 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" on page 679

Related tasks:

- "Authentication with groups and domain security (Windows)" on page 681
- "Installing DB2 on a backup domain controller" on page 680
- "Using a backup domain controller with DB2 database systems" on page 677

A scenario with server authentication (Windows)

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 and Windows security introduction" on page 675

A scenario with client authentication and a Windows 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 service however, it is not started automatically. To start the DB2 Security Service, enter the `NET START DB2NTSECSERVER` command.

Related concepts:

- "DB2 and Windows security introduction" on page 675

Support for global groups (on Windows)

DB2 database also supports global groups. In order to use global groups, you must include global groups inside a local group. When DB2 database 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" on page 679

Using a backup domain controller with DB2 database systems

Procedure:

If the server you use for DB2 database systems also acts as a backup domain controller, you can improve DB2 database performance and reduce network traffic if you configure DB2 database to use the backup domain controller.

You specify the backup domain controller to the DB2 database system by setting the DB2DMNBCKCTRL registry variable.

If you know the name of the domain for which DB2 database server is the backup domain controller, use:

```
db2dmnbckctlr=<domain_name>
```

where `domain_name` must be in upper case.

To have DB2 database system determine the domain for which the local machine is a backup domain controller, use:

```
DB2DMNBCKCTRL=?
```

Note: DB2 database 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 680

User authentication with DB2 for Windows

User authentication can cause problems for Windows users because of the way the operating system authenticates. This section describes some considerations for user authentication under DB2 for Windows:

- "User name and group name restrictions (Windows)"
- "DB2 database system and Windows security service" on page 680
- "Installing DB2 on a backup domain controller" on page 680
- "Authentication with groups and domain security (Windows)" on page 681

User name and group name restrictions (Windows)

The following are the limitations in this environment:

- User names and group names are limited to 30 characters within the DB2 database system.
- User names under Windows 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 the DB2 database. 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 database 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” on page 679
- “Trust relationships between domains on Windows” on page 679

Groups and user authentication on Windows

Users are defined on Windows by creating user accounts using the Windows administration tool called the “User Manager”.

An account containing other accounts, also called members, is a group. Groups give Windows 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 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:

- “Support for global groups (on Windows)” on page 677
- “Trust relationships between domains on Windows” on page 679

Related tasks:

- “Authentication with groups and domain security (Windows)” on page 681

Related reference:

- “User name and group name restrictions (Windows)” on page 678

Trust relationships between domains on Windows

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 a 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” on page 679
- “Support for global groups (on Windows)” on page 677

Related reference:

- “User name and group name restrictions (Windows)” on page 678

DB2 database system and Windows security service

In the DB2 database system, the authentication of user names and passwords is integrated with 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 and Windows security introduction” on page 675

Installing DB2 on a backup domain controller

Procedure:

In a Windows 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 server for Windows 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, the DB2 server 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 the DB2 server 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 server indicates that the system is a backup domain controller on the specified domain.

Under normal circumstances the setting `DB2DMNBCKCTRL=?` 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 the DB2 server to authenticate at the backup domain controller: set `DB2DMNBCKCTRL=xxx` where `xxx` is the Windows 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 database systems” on page 677

Authentication with groups and domain security (Windows)

Procedure:

The DB2 database system 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.

The DB2 database manager supports the following types of groups:

- Local groups
- Global groups
- Global groups as members of local groups.

The DB2 database manager enumerates the local and global groups that the user is a member of, using the security database where the user was found. The DB2 database system provides an override that forces group enumeration to occur on the local Windows server where the DB2 database 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 database 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 database tries to enumerate the user's groups on the local machine only. If the user is not defined as a member of a local or global group, then group enumeration fails. DB2 does **not** try to enumerate the user's groups on another machine in the domain or on the domain controllers.

If the DB2_GRP_LOOKUP profile registry variable is not set then:

1. The DB2 database system 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, the DB2 database system attempts to find the user name on its domain, and then on trusted domains.

If the DB2 database manager 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 the DB2 database manager is not running on a domain controller, then you should issue:

```
db2set -g DB2_GRP_LOOKUP=DOMAIN
```

This command tells the DB2 database system to use a domain controller in its own domain to find the name of a domain controller in the accounts domain. That is, when a DB2 database 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 the DB2 database is running on. There are two advantages to this method:

1. The nearest domain controller is found when the primary domain controller is unavailable.
2. The nearest domain controller is found when the primary domain controller is geographically remote.

Related concepts:

- "Acquiring Windows users' group information using an access token" on page 483
- "Groups and user authentication on Windows" on page 679

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 desktop is required in a Windows domain environment.

Related concepts:

- “DB2 and Windows security introduction” on page 675

Domain security support (Windows)

The following examples illustrate how the DB2 database management system can support Windows 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 94. Successful Connection Using a Domain Controller

Domain1	Domain2
A trust relationship exists with Domain2.	<ul style="list-style-type: none"> • 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.

Table 94. Successful Connection Using a Domain Controller (continued)

Domain1	Domain2
<p>The DB2 server runs in this domain. The following DB2 commands are issued from it:</p> <pre> REVOKE CONNECT ON db FROM public GRANT CONNECT ON db TO GROUP grp2 CONNECT TO db USER id2 </pre>	
<p>The local or global domain is scanned but id2 is not found. Domain security is scanned.</p>	
	<p>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).</p>
<p>The connection works because the user name and local or global group are on the same domain.</p>	

Related concepts:

- “Groups and user authentication on Windows” on page 679

Related tasks:

- “Authentication with groups and domain security (Windows)” on page 681

Appendix D. Using the Windows Performance Monitor

Windows performance monitor introduction

When working with DB2 database manager 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 database performance-related information.

- **DB2 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 computer operation including:

- CPU usage
- Memory utilization
- Disk activity
- Network activity

Related tasks:

- “Accessing remote DB2 database performance information” on page 688
- “Displaying DB2 database and DB2 Connect performance values” on page 687
- “Enabling remote access to DB2 performance information” on page 686
- “Registering DB2 with the Windows performance monitor” on page 685
- “Resetting DB2 performance values” on page 688

Related reference:

- “Windows performance objects” on page 687

Registering DB2 with the Windows performance monitor

Procedure:

The setup program automatically registers DB2 with the Windows Performance Monitor for you.

To make DB2 database 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 *Command Reference*

Enabling remote access to DB2 performance information

Procedure:

If your DB2 for Windows workstation is networked to other Windows computers, you can use the feature described in this section.

In order to see Windows performance objects from another DB2 for Windows computer, you must register an administrator username and password with the DB2 database manager. (The default Windows Performance Monitor username, **SYSTEM**, is a DB2 database reserved word and cannot be used.) To register the name, type:

```
db2perfr -r username password
```

Note: The username used must conform to the DB2 database 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 the DB2 database system, 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 database system does not match, local sessions of the Performance Monitor will not show DB2 database 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 663

Related reference:

- “db2perfr - Performance monitor registration tool command” in *Command Reference*

Displaying DB2 database and DB2 Connect performance values

Procedure:

To display DB2 database 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 computer, including “C:” and “D:”.

Related concepts:

- “Windows performance monitor introduction” on page 685

Related reference:

- “Windows performance objects” on page 687

Windows performance objects

Windows provides the following performance objects:

- **DB2 Database Manager**

This object provides general information for a single Windows instance. The DB2 database instance being monitored appears as the object instance.

For practical and performance reasons, you can only get performance information from one DB2 database instance at a time. The DB2 database instance that the Performance Monitor shows is governed by the db2instance registry variable in the Performance Monitor process. If you have multiple DB2 database 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 database instance to be monitored.

If you are running a partitioned database environment, you can only get performance information from one database partition server at a time. By default, the performance information for the default database partition (that is, the database partition that has logical port 0) is displayed. To see performance information of another database partition, you must start a separate session of the Performance Monitor with the DB2NODE environment variable set to the database partition number of the database partition to be monitored.

- **DB2 Databases**

This object provides information for a particular database. Information is available for each currently active database.

- **DB2 Applications**

This object provides information for a particular DB2 database application. Information is available for each currently active DB2 database 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 computer and what applications are active. For example, if the DB2 database manager is installed has been started, the DB2 Database Manager object will be listed. If there are also some DB2 databases and applications currently active on that computer, 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.

Accessing remote DB2 database 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 database or DB2 Connect code installed on that computer must be Version 6 or higher.

Related concepts:

- “Windows performance monitor introduction” on page 685

Resetting DB2 performance values

Procedure:

When an application calls the DB2 monitor APIs, the information returned is normally the cumulative values since the DB2 database 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 **db2perf** program. Type:
`db2perf`

By default, this resets performance values for all active DB2 databases. However, you can also specify a list of databases to reset. You can also use the `-d` option to specify that performance values for DCS databases should be reset. For example:

```
db2perf  
db2perf dbalias1 dbalias2 ... dbaliasn  
  
db2perf -d  
db2perf -d dbalias1 dbalias2 ... dbaliasn
```

The first example resets performance values for all active DB2 databases. The next example resets values for specific DB2 databases. The third example resets performance values for all active DB2 DCS databases. The last example resets values for specific DB2 DCS databases.

The **db2perf** program resets the values for ALL programs currently accessing database performance information for the relevant DB2 database server instance (that is, the one held in `DB2INSTANCE` in the session in which you run **db2perf**).

Invoking **db2perf** also resets the values seen by anyone remotely accessing DB2 database performance information when the **db2perf** command is executed.

Note: There is a DB2 database API, `sqlmrset`, that allows an application to reset the values it sees locally, not globally, for particular databases.

Related reference:

- “db2perf - Reset database performance values command” in *Command Reference*
- “db2ResetMonitor API - Reset the database system monitor data” in *Administrative API Reference*

Appendix E. DB2 Database technical information

Overview of the DB2 technical information

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
 - Topics
 - Help for DB2 tools
 - Sample programs
 - Tutorials
- DB2 books
 - PDF files (downloadable)
 - PDF files (from the DB2 PDF CD)
 - printed books
- Command line help
 - Command help
 - Message help
- Sample programs

IBM periodically makes documentation updates available. If you access the online version on the DB2 Information Center at ibm.com[®], you do not need to install documentation updates because this version is kept up-to-date by IBM. If you have installed the DB2 Information Center, it is recommended that you install the documentation updates. Documentation updates allow you to update the information that you installed from the *DB2 Information Center CD* or downloaded from Passport Advantage as new information becomes available.

Note: The DB2 Information Center topics are updated more frequently than either the PDF or the hard-copy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 Information Center at ibm.com.

You can access additional DB2 technical information such as technotes, white papers, and Redbooks™ online at ibm.com. Access the DB2 Information Management software library site at <http://www.ibm.com/software/data/sw-library/>.

Documentation feedback

We value your feedback on the DB2 documentation. If you have suggestions for how we can improve the DB2 documentation, send an e-mail to db2docs@ca.ibm.com. The DB2 documentation team reads all of your feedback, but cannot respond to you directly. Provide specific examples wherever possible so that we can better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use this e-mail address to contact DB2 Customer Support. If you have a DB2 technical issue that the documentation does not resolve, contact your local IBM service center for assistance.

Related concepts:

- “Features of the DB2 Information Center” in *Online DB2 Information Center*
- “Sample files” in *Samples Topics*

Related tasks:

- “Invoking command help from the command line processor” in *Command Reference*
- “Invoking message help from the command line processor” in *Command Reference*
- “Updating the DB2 Information Center installed on your computer or intranet server” on page 697

Related reference:

- “DB2 technical library in PDF format” on page 692

DB2 technical library in PDF format

The following tables describe the DB2 library available from the IBM Publications Center at www.ibm.com/shop/publications/order.

Although the tables identify books available in print, the books might not be available in your country or region.

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 or other DB2 products.

Table 95. DB2 technical information

Name	Form Number	Available in print
<i>Administration Guide: Implementation</i>	SC10-4221	Yes
<i>Administration Guide: Planning</i>	SC10-4223	Yes
<i>Administrative API Reference</i>	SC10-4231	Yes
<i>Administrative SQL Routines and Views</i>	SC10-4293	No
<i>Call Level Interface Guide and Reference, Volume 1</i>	SC10-4224	Yes
<i>Call Level Interface Guide and Reference, Volume 2</i>	SC10-4225	Yes
<i>Command Reference</i>	SC10-4226	No
<i>Data Movement Utilities Guide and Reference</i>	SC10-4227	Yes
<i>Data Recovery and High Availability Guide and Reference</i>	SC10-4228	Yes
<i>Developing ADO.NET and OLE DB Applications</i>	SC10-4230	Yes
<i>Developing Embedded SQL Applications</i>	SC10-4232	Yes
<i>Developing SQL and External Routines</i>	SC10-4373	No

Table 95. DB2 technical information (continued)

Name	Form Number	Available in print
<i>Developing Java Applications</i>	SC10-4233	Yes
<i>Developing Perl and PHP Applications</i>	SC10-4234	No
<i>Getting Started with Database Application Development</i>	SC10-4252	Yes
<i>Getting started with DB2 installation and administration on Linux and Windows</i>	GC10-4247	Yes
<i>Message Reference Volume 1</i>	SC10-4238	No
<i>Message Reference Volume 2</i>	SC10-4239	No
<i>Migration Guide</i>	GC10-4237	Yes
<i>Net Search Extender Administration and User's Guide</i> Note: HTML for this document is not installed from the HTML documentation CD.	SH12-6842	Yes
<i>Performance Guide</i>	SC10-4222	Yes
<i>Query Patroller Administration and User's Guide</i>	GC10-4241	Yes
<i>Quick Beginnings for DB2 Clients</i>	GC10-4242	No
<i>Quick Beginnings for DB2 Servers</i>	GC10-4246	Yes
<i>Spatial Extender and Geodetic Data Management Feature User's Guide and Reference</i>	SC18-9749	Yes
<i>SQL Guide</i>	SC10-4248	Yes
<i>SQL Reference, Volume 1</i>	SC10-4249	Yes
<i>SQL Reference, Volume 2</i>	SC10-4250	Yes
<i>System Monitor Guide and Reference</i>	SC10-4251	Yes
<i>Troubleshooting Guide</i>	GC10-4240	No
<i>Visual Explain Tutorial</i>	SC10-4319	No
<i>What's New</i>	SC10-4253	Yes
<i>XML Extender Administration and Programming</i>	SC18-9750	Yes
<i>XML Guide</i>	SC10-4254	Yes
<i>XQuery Reference</i>	SC18-9796	Yes

Table 96. DB2 Connect-specific technical information

Name	Form Number	Available in print
<i>DB2 Connect User's Guide</i>	SC10-4229	Yes
<i>Quick Beginnings for DB2 Connect Personal Edition</i>	GC10-4244	Yes

Table 96. DB2 Connect-specific technical information (continued)

Name	Form Number	Available in print
<i>Quick Beginnings for DB2 Connect Servers</i>	GC10-4243	Yes

Table 97. WebSphere Information Integration technical information

Name	Form Number	Available in print
<i>WebSphere Information Integration: Administration Guide for Federated Systems</i>	SC19-1001	Yes
<i>WebSphere Information Integration: ASNCLP Program Reference for Replication and Event Publishing</i>	SC19-1000	Yes
<i>WebSphere Information Integration: Configuration Guide for Federated Data Sources</i>	SC19-1034	No
<i>WebSphere Information Integration: SQL Replication Guide and Reference</i>	SC19-1002	Yes

Note: The DB2 Release Notes provide additional information specific to your product's release and fix pack level. For more information, see the related links.

Related concepts:

- "Overview of the DB2 technical information" on page 691
- "About the Release Notes" in *Release notes*

Related tasks:

- "Ordering printed DB2 books" on page 694

Ordering printed DB2 books

If you require printed DB2 books, you can buy them online in many but not all countries or regions. You can always order printed DB2 books from your local IBM representative. Keep in mind that some softcopy books on the *DB2 PDF Documentation CD* are unavailable in print. For example, neither volume of the *DB2 Message Reference* is available as a printed book.

Printed versions of many of the DB2 books available on the DB2 PDF Documentation CD can be ordered for a fee from IBM. Depending on where you are placing your order from, you may be able to order books online, from the IBM Publications Center. If online ordering is not available in your country or region, you can always order printed DB2 books from your local IBM representative. Note that not all books on the DB2 PDF Documentation CD are available in print.

Note: The most up-to-date and complete DB2 documentation is maintained in the DB2 Information Center at <http://publib.boulder.ibm.com/infocenter/db2help/>.

Procedure:

To order printed DB2 books:

- To find out whether you can order printed DB2 books online in your country or region, check the IBM Publications Center at <http://www.ibm.com/shop/publications/order>. You must select a country, region, or language to access publication ordering information and then follow the ordering instructions for your location.
- To order printed DB2 books from your local IBM representative:
 - Locate the contact information for your local representative from one of the following Web sites:
 - The IBM directory of world wide contacts at www.ibm.com/planetwide
 - The IBM Publications Web site at <http://www.ibm.com/shop/publications/order>. You will need to select your country, region, or language to the access appropriate publications home page for your location. From this page, follow the "About this site" link.
 - When you call, specify that you want to order a DB2 publication.
 - Provide your representative with the titles and form numbers of the books that you want to order.

Related concepts:

- "Overview of the DB2 technical information" on page 691

Related reference:

- "DB2 technical library in PDF format" on page 692

Displaying SQL state help from the command line processor

DB2 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 command help from the command line processor" in *Command Reference*
- "Invoking message help from the command line processor" in *Command Reference*

Accessing different versions of the DB2 Information Center

For DB2 Version 9 topics, the DB2 Information Center URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>.

For DB2 Version 8 topics, go to the Version 8 Information Center URL at: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>.

Related tasks:

- “Setting up access to DB2 contextual help and documentation” on page 435

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. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

To display topics in your preferred language in a Firefox or Mozilla browser:

1. Select the **Tools** —> **Options** —> **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. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

On some browser and operating system combinations, you might have to also change the regional settings of your operating system to the locale and language of your choice.

Related concepts:

- “Overview of the DB2 technical information” on page 691

Updating the DB2 Information Center installed on your computer or intranet server

If you have a locally-installed DB2 Information Center, updated topics can be available for download. The 'Last updated' value found at the bottom of most topics indicates the current level for that topic.

To determine if there is an update available for the entire DB2 Information Center, look for the 'Last updated' value on the Information Center home page. Compare the value in your locally installed home page to the latest value which is available on the IBM hosted Information Center home page. If they are the same, you have the latest documentation level and no update is required. If they are not the same, you should update your locally-installed Information Center.

Updating your locally-installed DB2 Information Center requires that you:

1. Stop the DB2 Information Center on your computer, and restart the Information Center in stand-alone mode. Running the Information Center in stand-alone mode prevents other users on your network from accessing the Information Center, and allows you to download and apply updates.
2. Use the Update feature to determine if update packages are available from IBM. If update packages are available, use the Update feature to download the packages. (The Update feature is only available in stand-alone mode.)
3. Stop the stand-alone Information Center, and restart the DB2 Information Center service on your computer.

Procedure:

To update the DB2 Information Center installed on your computer or intranet server:

1. Stop the DB2 Information Center service.
 - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Stop**.
 - On Linux, enter the following command:
`/etc/init.d/db2icdv9 stop`
2. Start the Information Center in stand-alone mode.
 - On Windows:
 - a. Open a command window.
 - b. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the C:\Program Files\IBM\DB2 Information Center\Version 9 directory.
 - c. Run the `help_start.bat` file using the fully qualified path for the DB2 Information Center:
`<DB2 Information Center dir>\doc\bin\help_start.bat`
 - On Linux:
 - a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the `/opt/ibm/db2ic/V9` directory.
 - b. Run the `help_start.sh` file using the fully qualified path for the DB2 Information Center:
`<DB2 Information Center dir>/doc/bin/help_start`

The systems default Web browser launches to display the stand-alone Information Center.

3. Click the Update button (🔄). On the right hand panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the download process, check the selections you want to download, then click **Install Updates**.
5. After the download and installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center.
 - On Windows, run the `help_end.bat` file using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>\doc\bin\help_end.bat
```
 - On Linux, run the `help_end.sh` file using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>/doc/bin/help_end
```
7. Restart the DB2 Information Center service.
 - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Start**.
 - On Linux, enter the following command:

```
/etc/init.d/db2icdv9 start
```

The updated DB2 Information Center displays the new and updated topics.

Related concepts:

- “DB2 Information Center installation options” in *Quick Beginnings for DB2 Servers*

Related tasks:

- “Installing the DB2 Information Center using the DB2 Setup wizard (Linux)” in *Quick Beginnings for DB2 Servers*
- “Installing the DB2 Information Center using the DB2 Setup wizard (Windows)” in *Quick Beginnings for DB2 Servers*

DB2 Visual Explain tutorial

The DB2 Visual Explain tutorial helps you learn about analyzing, optimizing, and tuning SQL statements for better performance. Lessons provide step-by-step instructions.

Before you begin:

You can view the XHTML version of the tutorial from the Information Center at <http://publib.boulder.ibm.com/infocenter/db2help/>.

Some lessons use sample data or code. See the tutorial for a description of any prerequisites for its specific tasks.

DB2 Visual Explain tutorial:

To view the tutorial, click on the title.

Visual Explain Tutorial

Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

Related concepts:

- “Visual Explain overview” on page 451

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 in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

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 or bug fixes), fix packs, 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/support.html>

Related concepts:

- “Introduction to problem determination” in *Troubleshooting Guide*
- “Overview of the DB2 technical information” on page 691

Terms and Conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal use: You may reproduce these Publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these Publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Appendix F. 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

Company, product, or service names identified in the documents of the DB2 Version 9 documentation library may be trademarks or service marks of International Business Machines Corporation or other companies. Information on the trademarks of IBM Corporation in the United States, other countries, or both is located at <http://www.ibm.com/legal/copytrade.shtml>.

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 documentation library:

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

Intel[®], Itanium[®], Pentium[®], and Xeon[®] 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.

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

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

Index

Special characters

\$RAHBUFDIR 133
\$RAHBUFNAME 133
\$RAHENV 139

A

access

- label-based access control (LBAC) 538
- to DB2 information Center 435

access control

- authentication 490
- column-specific 538
- database manager 519
- database objects 519
- row-specific 538
- view to table 525

access plan graph 453

access plan graph node 454

access plans

- creating from Command Editor 465
- out-of-date 467
- overview 452
- retrieving when using LONGDATACOMPAT 468
- viewing graphical presentation 473

access token 483

active directory

- configuring DB2 576
- DB2 objects 593
- extending the directory schema 591
- Lightweight Directory Access Protocol (LDAP) 573
- security 590
- support 575

adding

- automatic prefetchsize adjustment 288
- database partitions 125
- foreign keys 310
- primary keys 309
- table check constraints 314
- unique constraints 313

administration log file 182

administration server 91

administration tools

- service-level information 370
- shutting down 369

administrative views

- AUTHORIZATIONIDS 610, 613
- DB_HISTORY 87
- OBJECTOWNERS 613
- PRIVILEGES 610, 613

advisors

- using 385

aggregating function 243

AIX

- large page support 12

AIX (continued)

- system commands
 - vmo 12
 - vmtune 12

alert objects

- viewing 447

alert summaries

- DB2 Health Monitor 62
- viewing 446

aliases

- authority 254
- creating 254
- DB2 for z/OS and OS/390 254
- dropping 332
- using 254

ALTER COLUMN clause

- in table columns 305

alter materialized query table

- properties 335

ALTER privilege 515

ALTER TABLE statement 300

- adding check constraint example 314
- adding columns example 304
- adding keys example 310
- adding unique constraint example 313
- dropping check constraint example 317
- dropping keys example 316
- dropping unique constraint example 315

ALTER TABLESPACE statement

- example of 285

ALTER VIEW statement

- example 330

altering

- columns 305
- database partition group 281
- IDENTITY column 308
- structured type 333
- table spaces 284
- views 330

altering a table 295

altering tables 297

- using ALTER TABLE statement 300
- using stored procedures 324

alternate servers

- examples 49
- identifying 49

application programming interfaces (API)

- updating database directories 180

ATTACH command 5

- attaching data partitions
 - description 346

attribute definitions

- Netscape LDAP 593

attributes

- table, changing 298

audit activities 621

audit facility

- actions/events 621

audit facility (continued)

- asynchronous record writing 623
- audit data in tables
 - creating audit data files 631
 - creating tables for audit data 628
 - loading tables with audit data 632
 - overview 628
 - selecting data from tables 635
- audit events table 637
- authorities/privileges 621
- behavior 623
- CHECKING access approval reasons 640
- CHECKING access attempted types 641
- checking events table 638
- CONTEXT audit events 653
- CONTEXT events table 652
- controlling activities 655
- error handling 623
- ERRORTYPE parameter 623
- examples 655
- messages 636
- monitoring access to data 527
- OBJMAINT events table 643
- parameter descriptions 624
- record layouts 636
- SECMAINT events table 645
- SECMAINT privileges or authorities 647
- synchronous record writing 623
- syntax 624
- SYSADMIN audit events 650
- SYSADMIN events table 650
- tips and techniques 654
- usage scenarios 624
- VALIDATE events table 651

audit records

- object types 639

audit trail 621

audit_buf_sz configuration

- parameter 623

authentication

- definition of 490
- domain security 681
- groups 681
- Kerberos
 - details 496
- partitioned database considerations 496
- remote client 495
- types
 - CLIENT 490
 - KERBEROS 490
 - KRB_SERVER_ENCRYPT 490
 - SERVER 490
 - SERVER_ENCRYPT 490
- using an ordered domain list 682

- authority levels
 - database administration (DBADM) 509, 513
 - removing DBADM from SYSADM 506
 - removing DBADM from SYSCTRL 507
 - security administrator (SECADM) 508
 - See privileges 501
 - system administration (SYSADM) 506
 - system control (SYSCTRL) 507
 - system maintenance (SYSMAINT) 508
 - system monitor authority (SYSMON) 510
- authorization
 - trusted client 490
- authorization ID
 - changing
 - SETSESSIONUSER 513
- authorization names
 - create view for privileges
 - information 613
 - retrieving for privileges
 - information 610
 - retrieving names with DBADM
 - authority 611
 - retrieving names with table access
 - authority 612
 - retrieving privileges granted to 613
- authorized user infraction information
 - viewing 415
- Autoconfigure API 84
- AUTOCONFIGURE command 84
 - sample output 85
- automatic client reroute
 - configuration 52
 - connection failures 53
 - description 45
 - examples 49
 - limitations 47
 - roadmap 45
 - setup 45
- automatic prefetch size adjustment
 - after adding or dropping
 - containers 288
- automatic storage
 - for databases 54
 - restrictions 62
 - table spaces 58
 - automatic and large 63
 - temporary for table spaces 57
- automatic storage path
 - adding 64
- automatic summary tables
 - creating 201

B

- backing up data
 - using the Backup wizard 387
- backup domain controller
 - configuring DB2 677
 - installing DB2 680

- Backup wizard
 - backing up data 387
- backups
 - security risks 616
- BIND command
 - OWNER option 523
- bind options
 - viewing SQL or XQuery statement
 - details 469
- BIND privilege
 - definition 517
- BINDADD database authority
 - definition 511
- binding
 - database utilities 183
 - rebinding invalid packages 521
- block-structured devices 149
- boundary ranges
 - specifying 195
- buffer pools
 - altering 283
 - creating 166
 - for partitioned databases 167
- built-in functions
 - viewing SQL or XQuery statement
 - details 469

C

- caching
 - file system for table spaces 159
- call level interface (CLI)
 - binding to a database 183
- CARD row
 - under Statistics column 469
- CASCADE semantic
 - for DROP COLUMN 300
- CATALOG DATABASE command
 - example 176
- catalog nodes 9
- catalog tables
 - stored on database catalog node 9
- cataloging
 - database systems 177
- categories, task
 - managing 431
- character serial devices 149
- character strings
 - data type 217
- check constraints
 - adding 229, 314
 - adding or changing 297
 - changing 314
 - defining 228
 - dropping 317
- CLIENT authentication type
 - client-level security 490
- client communication errors 45
- client connectivity
 - using multiple DB2 copies
 - on Windows 22
- client reroute
 - automatic 45
 - examples 49
 - interaction with connection
 - timeout 52
 - JCC Type 4 drivers 54
- client reroute (*continued*)
 - limitations 47
- clients
 - automatic rerouting 44
- clustering
 - definition 454
- code sets
 - success 420
 - managing 430
- code sets, success
 - managing 430
- column distribution
 - viewing SQL or XQuery statement
 - statistics 469
- column groups
 - viewing SQL or XQuery statement
 - details 469
- column properties
 - changing 304
- column UDFs 243
- columns
 - adding or changing 297
 - defining 217
 - definition
 - modifying 305
 - dropping LBAC protected 569
 - dropping or removing 320
 - effect of LBAC on reading 560
 - inserting LBAC protected 563
 - protecting a column with LBAC 558
 - updating LBAC protected 565
- Command Editor
 - adding access plans 465
 - options, setting 449
- command line processor (CLP)
 - binding to a database 183
- command statement
 - setting termination character 434
- commands
 - running in parallel 133
- compliance details
 - resetting 415
 - viewing 415
- compression
 - data row 188
 - existing tables 295
 - new tables 187
 - row 188
- concepts
 - scheduler 420
 - success code sets 420
- Configuration Advisor
 - generating recommended values 84
 - sample output 85
- configuration parameters
 - partitioned database 9
 - TCP_KEEPALIVE 53
 - viewing SQL or XQuery statement
 - details 469
- configuring
 - LDAP 576
 - LDAP user for applications 578
- CONNECT database authority 511
- connection failures
 - automatic client reroute 53

- connectivity
 - client, using multiple DB2 copies
 - on Windows 22
- connectTimeout
 - interaction with client reroute 52
- constraint violations
 - checking
 - using the SET INTEGRITY statement 230
- constraints
 - check
 - adding 229
 - changing 314
 - defining
 - foreign keys 226
 - referential constraints 224
 - unique constraints 223
 - dropping
 - unique constraints 315
 - informational 233
 - table check 228
- contact groups
 - managing 428
- contacting IBM 707
- contacts
 - managing 428
- containers
 - adding to SMS table spaces 289
 - definition 455
 - DMS table spaces
 - adding containers to 285
 - modifying containers in 286
- contextual help
 - setting up access 435
 - settings for DB2 administration tools 435
- Control Center
 - DB2 federated system objects
 - adding to the object tree 389
 - displaying objects 392
 - displaying table information in the contents pane 392
 - extensions
 - adding a folder 403
 - adding an example object 405
 - adding an object 407
 - adding the remove action 409
 - altering an object 410
 - creating sub menus 401
 - guidelines for plug-in developers 395
 - plug-in architecture 395
 - positioning the menu item 400
 - writing plug-ins 397
 - filtering or pre-filtering objects 394
 - finding objects in the contents pane 394
 - Help
 - accessing 435
 - how to access Help 385
 - legend 380
 - managing database partitions 282
 - obtaining diagnostic information 393
 - opening 382
 - overview 376
 - showing related objects 270

- CONTROL privilege
 - described 515
 - implicit issuance 522
 - package privileges 517
- controlling the rah command 139
- cooked devices 149
- copy schema
 - operation, restarting 173
- copying
 - schemas and objects 170
 - tables 296
- cost
 - definition 455
- CREATE ALIAS statement
 - example of 254
- CREATE DATABASE command
 - example of 113
 - RESTRICTIVE option 613
- CREATE INDEX statement
 - examples 261
 - online reorganization 258, 261
 - overview 467
 - restrict access 261
 - unique index 261
- CREATE TABLE statement 189
 - defining check constraints 228
 - defining referential constraints 224
 - example of 217
 - using multiple table spaces 190
- CREATE TABLESPACE statement
 - example of 149
- CREATE TRIGGER statement
 - example of 240
- CREATE VIEW statement
 - changing column names 251
 - CHECK OPTION clause 251
 - example of 251
- CREATE_EXTERNAL_ROUTINE
 - database authority 511
- CREATE_NOT_FENCED_ROUTINE
 - database authority 511
- CREATETAB database authority 511
- creating
 - aliases 254
 - buffer pools
 - for partitioned databases 167
 - database objects 382
 - function mappings 244
 - function templates 245
 - hierarchy tables 216
 - index extensions 258
 - index specifications 258
 - indexes 467
 - enabling parallelism 10
 - overview 256
 - instances
 - UNIX 39
 - Windows 40
 - LBAC security labels 547
 - LDAP users 577
 - schemas 168
 - table spaces 149
 - tables 217
 - using a wizard 190
 - tables in multiple table spaces 190
 - tasks 425
 - triggers 240

- creating (*continued*)
 - type mappings 248
 - typed tables 216
 - user-defined distinct types 247
 - user-defined functions 243
 - user-defined types 246
 - views 251
- CURRENT SCHEMA special register 6, 169
- cursor blocking
 - definition 460
- custom folders
 - deleting 391

D

- DAS (DB2 Administration Server)
 - first failure data capture 111
 - Java virtual machine setup 101
 - setting up when running multiple DB2 copies 24
- data
 - audit
 - creating audit data files 631
 - creating tables for 628
 - loading audit data into tables 632
 - selecting audit data from tables 635
 - working with, overview 628
 - changing distribution 281
 - controlling database access 481
 - effect of LBAC on reading 560
 - indirect access 616
 - inserting LBAC protected 563
 - monitoring access 527
 - protecting with LBAC 558
 - securing system catalog 613
 - updating LBAC protected 565
- data encryption
 - description 527
- data movement mode
 - changing for tables 238
- data partitions
 - adding 356
 - attach 210
 - attach, rolling-in data 336
 - attached 338
 - attaching 339
 - attributes 354
 - creating 195
 - defining the range 195
 - detach, rolling-out data 336
 - detached 338
 - detaching 339
 - dropping 358
 - rolling in, attaching 342
 - rolling out, detaching 342
 - rolling-in data, attaching 346
 - rolling-out, detaching 352
 - specifying 195
- data redistribution 128
- data row compression 188
- data types
 - column definition 217
 - multibyte character set 217
 - source 249
 - length limits 250

- database 3
 - before creating 3
 - changing 281
 - considerations before changing 275
 - considerations for creating 33
 - creating 113
- database access
 - controlling 481
- database administration (DBADM)
 - authority
 - description 509
- database authorities
 - BINDADD 511
 - CONNECT 511
 - CREATE_EXTERNAL_ROUTINE 511
 - CREATE_NOT_FENCED 511
 - CREATETAB 511
 - database manager (DBADM) 511
 - granting 511
 - granting to new groups 529
 - granting to new users 529
 - IMPLICIT_SCHEMA 511
 - LOAD 511
 - PUBLIC 511
 - QUIESCE_CONNECT 511
 - revoking 511
 - SECADM 511
 - security administrator (SECADM) 511
- database configuration
 - changing 279
 - changing across partitions 281
- database configuration file
 - creating 80
- database configuration parameters
 - generating recommended values 84
- database directories
 - updating 180
- database manager
 - access control 519
 - binding utilities 183
 - configuration parameters
 - generating recommended values 84
 - index 256
 - starting on UNIX 4
 - starting on Windows 4
 - stopping on UNIX 13
 - stopping on Windows 14
- database objects
 - access control 519
 - creating 382
 - modifying
 - statement dependencies 366
 - naming rules
 - NLS 668
 - Unicode 669
- database partition groups
 - altering 281
 - creating 115
 - distribution key, changing 318
 - IBMDEFAULTGROUP default table 191
 - initial definition 115
 - table considerations 191
- database partition number 81
- database partition servers
 - description 30
 - dropping 147
 - issuing commands 130
 - specifying 138
 - Windows 143
- database partitions
 - adding 123, 125
 - to a running system 119
 - to a stopped system 120
 - to NT system 122
 - adding using a wizard 124
 - cataloging 9, 179
 - changing 145
 - changing database configuration 281
 - changing in database partition group 281
 - creating database across all 9
 - dropping from an instance 127
 - managing 116
 - from the Control Center 282
- database recovery log
 - defining 182
- database server capacity
 - methods of expanding 29
- database servers
 - alternate 49
- database systems
 - cataloging 177
- databases
 - access
 - privileges through package with queries 523
 - altering database partition group 281
 - automatic storage 54
 - cataloging 176
 - changing distribution of data 281
 - creating across all database partitions 9
 - directory information, changing 180
 - dropping 293
 - enabling data partitioning 9
 - enabling I/O parallelism 11
 - label-based access control (LBAC) 538
 - package dependencies 366
 - quiescing 186
 - restore implications 59
 - unavailable status 391
 - unquiescing 186
- DB_HISTORY administrative view 87
- DB2 Administration Server (DAS)
 - configuration 106
 - configuring 95
 - creating 93
 - enabling discovery 107
 - listing 95
 - notification and contact list setup 100
 - overview 91
 - ownership rules 79
 - removing 103
 - scheduler setup and configuration 96
 - security considerations 102
 - setting up with partitioned database system 104
 - example 104
- DB2 Administration Server (DAS)
 - (continued)
 - starting and stopping 94
 - update configuration 110
 - updating on UNIX 102
 - using Configuration Assistant and Control Center 110
- DB2 administration tools
 - setting hover help 435
- DB2 Administration Tools
 - starting 369
- DB2 copies
 - changing the default copy after installation 21
 - differences 17
 - managing 26
 - restrictions 17
 - roadmap 15
 - setting the DAS 24
 - setting the default instance 25
 - uninstalling 28
 - using on same computer 17
- DB2 database help
 - using 370
- DB2 environment
 - automatically set
 - UNIX 43
 - manually set
 - UNIX 44
- DB2 for Windows Performance Counters 685
- DB2 Health Monitor
 - alert summaries 62
- DB2 Help menu 375
- DB2 information Center
 - setting up access 435
- DB2 Information Center
 - updating 697
 - versions 695
 - viewing in different languages 696
- DB2 objects
 - naming rules 663
- DB2 tools catalog
 - creating a database for 424
- DB2 Tools menu 374
- DB2 UDB for z/OS health monitor
 - overview 441
 - starting, stopping, refreshing 442
 - viewing alert objects 447
 - viewing alert summaries 446
 - viewing, submitting, saving recommended actions 443
- DB2 Universal JDBC driver
 - client reroute support 45
- db2_all command 130, 131, 132
 - overview 130
- db2_call_stack 131
- DB2_CONNRETRIES_INTERVAL 70
 - registry variable 52
- DB2_INDEX_TYPE2 70
- db2_kill 131
- DB2_LIC_STAT_SIZE 70
- DB2_MAX_CLIENT_CONNRETRIES
 - registry variable 52
- DB2_MAX_CLIENT_CONNRETRIES 70
- DB2_VIEW_REOPT_VALUES 70
- DB2_WORKLOAD 75

- DB2ACCOUNT 70
- DB2ADMNS 486
- db2audit 624
- db2audit.log 621
- DB2BIDI 70
- DB2CODEPAGE 70
- DB2DBDFT 70
- DB2DBMSADDR 70
- db2diag.log 182
- DB2DISCOVERYTIME 70
- DB2DMNBCKCTLR profile registry variable 677, 680
- db2gncol utility 321
- DB2GRAPHICUNICODESERVER 70
- db2icrt command
 - creating additional instances 38
- db2idrop command 278
- db2ilist command 41
- DB2INCLUDE 70
- DB2INSTANCE environment variable
 - defining default instance 16
- DB2INSTDEF 70
- DB2INSTOWNER 70
- db2iupdt command 276, 277
- DB2LBACREADARRAY rule 552
- DB2LBACREADSET rule 552
- DB2LBACREADTREE rule 552
- DB2LBACRULES LBAC rule set 552
- DB2LBACWRITEARRAY rule 552
- DB2LBACWRITESET rule 552
- DB2LBACWRITETREE rule 552
- DB2LDAP_CLIENT_PROVIDER 575
- db2ldcfg utility 578
- DB2LOCALE 70
- DB2NBDISCOVERRCVBUFS 70
- db2nchg command 145
- db2ncrt command 144
- db2ndrop command 147
- db2nlist command 143
- DB2NODE
 - exported when adding server 119, 120, 122
- db2nodes.cfg file 81
- DB2NTNOCACHE option 159
- db2perfc command 688
- db2perfi command 685
- db2perfr command 686
- DB2SECURITYLABEL
 - providing explicit values 557
 - viewing as a string 557
- db2set command 65, 68
- db2start ADDNODE 144
- db2start command 4
- db2stop command 13, 14
- DB2TERRITORY 70
- DB2USERS 486
- DBADM (database administration)
 - authority
 - description 509
- DBADM authority
 - retrieving names 611
- DBCS (double-byte character set)
 - naming rules 668
- DDL statements
 - generating 183
- DECLARE GLOBAL TEMPORARY TABLE 212

- default
 - notification message 423
- default attribute specification 217
- default DB2 copy
 - changing after installation 21
- default options
 - setting 436
- default scheduling scheme
 - setting 449
- DELETE privilege 515
- design, implementing 3
- DETACH command
 - overview of 5
- detached data partitions 354
 - description 352
- diagnostic information
 - obtaining
 - in the Control Center 393
- dimensions
 - defining on a table 235
- Direct I/O (DIO)
 - supported configuration 159
- directories
 - local database directory 178
 - system database directory 178
 - updating 180
- directory information
 - database, changing 180
- directory schema
 - extending
 - for IBM SecureWay Directory Server 595
 - for Sun One Directory Server 596
- directory support
 - Netscape LDAP 593
- disabling notification
 - using the Health Center Status Beacon 448
- discovery feature
 - configuration 110
 - enabling 107
 - hiding server instances 108
 - setting parameters 109
- distribution keys
 - changing 318
 - defining 229
 - index distributed on 258
 - table considerations 191
- DMS table spaces
 - creating 149
- documentation 691, 692
 - terms and conditions of use 699
- domain controller
 - backup 677
- domain list
 - ordered 682
- domain security
 - authentication 681
 - Windows support 683
- domains
 - trust relationships 679
- DROP DATABASE command
 - example 293
- Drop Database Partitions launchpad 127
- DROP statement
 - indexes 327
 - table spaces 291

- DROP statement (*continued*)
 - tables
 - examples 363
 - views
 - examples 330
- dropped table recovery
 - CREATE TABLESPACE statement 149
- dropping
 - aliases 332
 - columns 300, 320
 - LBAC protected 569
 - containers 288
 - database partitions using a launchpad 127
 - databases 293
 - foreign keys 316
 - index extensions 327
 - index specifications 327
 - indexes 327
 - LBAC security labels 547
 - materialized query tables 365
 - primary keys 316
 - schemas 294
 - sequences 320
 - staging tables 365
 - table check constraints 317
 - tables 363
 - triggers 329
 - type mappings 334
 - unique constraints 315
 - user table spaces 291
 - user-defined functions 333
 - user-defined tables 364
 - user-defined types 334
 - views 330
- dynamic SQL or XQuery statements
 - cached
 - marked invalid 327
 - definition 455
 - EXECUTE privilege for database access 523

E

- eliminating duplicate machine entries 138
- enabling notification
 - using the Health Center Status Beacon 448
- encrypting data 527
- Enhanced DIO
 - supported configuration 159
- Enhanced Journal File System (JFS2) 159
- environment variables
 - declaring 68
 - profile registry 65
 - rah 139
 - RAHDOTFILES 140
 - setting
 - UNIX 79
 - Windows 77
- environment-specific information 371
- error messages
 - when adding nodes to partitioned databases 128

- examples
 - alternate server 49
 - automatic client reroute 49
- EXECUTE privilege
 - database access with dynamic queries 523
 - database access with static queries 523
 - definition 517, 518
- explain snapshot
 - definition 456
- explain tables
 - creating 466
- explainable statements
 - definition 456
 - viewing 474
- explained SQL statements
 - definition 456
 - viewing history 476
- explained XQuery statements
 - definition 456
 - viewing history 476
- explicit schema use 6
- expressions
 - NEXTVAL 234
 - PREVVAL 234
- extended security
 - Windows 486

F

- fast communications manager (FCM)
 - service entry syntax 32
- FCM (fast communications manager)
 - service entry syntax 32
- federated databases
 - function mapping, creating 244
 - function template, creating 245
 - index specification
 - creating 258
 - object naming rules 666
 - type mapping, creating 248
- federated systems objects
 - adding to the Control Center object tree 389
- file system caching
 - for table spaces 159
- filtering
 - objects in the Control Center 394
- finding
 - objects
 - in the Control Center contents pane 394
- firewalls
 - application proxy 620
 - circuit level 620
 - description 619
 - screening router 619
 - stateful multi-layer inspection (SMLI) 620
- first failure data capture (FFDC)
 - on DAS 111
- fonts
 - changing for menus and text 437
- foreign key constraints
 - referential constraints 226
 - rules for defining 226

- foreign keys
 - adding or changing 297
 - adding to a table 310
 - changing 311
 - composite 226
 - constraint name 226
 - DROP FOREIGN KEY clause, ALTER TABLE statement 316
 - import utility, referential integrity
 - implications for 227
 - load utility, referential integrity
 - implications for 227
 - privileges required for dropping 316
 - rules for defining 226
- format
 - security label as string 549
- function invocation, selectivity 268
- function mappings
 - creating 244
- function privileges 518
- function statistics
 - viewing SQL or XQuery statement statistics 469
- function templates
 - creating 245
- functions
 - DECRYPT 527
 - dropping a user-defined 333
 - ENCRYPT 527
 - GETHINT 527

G

- generated columns
 - defining on a new table 219
 - modifying 308
- global group support
 - Windows 677
- global level profile registry 65
- GRANT statement
 - example 519
 - implicit issuance 522
 - use of 519
- granting
 - LBAC security labels 547
- granting database authorities
 - to new groups 529
 - to new users 529
- granting privileges
 - to new groups 530
 - to new users 534
- group information
 - access token 483
- grouping tasks 425
- groups
 - naming rules 666
 - selecting 481
 - selecting new tasks for 427
- groups and user authentication
 - Windows 679
- guidelines
 - range-clustered tables 216

H

- Health Center Status Beacon
 - enabling or disabling notification 448
- help
 - accessing 435
 - displaying 696
 - for DB2 administration tools 370
 - for SQL statements 695
 - how to access
 - in the Control Center 385
- hierarchy tables
 - creating 216
 - dropping 363
- historical information
 - viewing in Journal 418
- history file
 - accessing 87
- hover help
 - setting for DB2 administration tools 435

I

- I/O parallelism
 - enabling 11
- IBM eNetwork Directory, object classes and attributes 598
- IBM SecureWay Directory Server
 - extending the directory schema for 595
- IBMCATGROUP database partition
 - group 115
- IBMDEFAULTGROUP database partition
 - group 115
- IBMTMPGROUP database partition
 - group 115
- identity columns
 - altering 318
 - defining on a new table 220
 - modifying 308
- IDENTITY columns 235
 - modifying 308
- implicit authorization
 - managing 522
- implicit schema authority
 - (IMPLICIT_SCHEMA) 513
- implicit schema use 6
- IMPLICIT_SCHEMA
 - authority 168
 - database authority 511
- IMS
 - setting options 450
- index extension 258
- index keys 258
- index maintenance
 - details 266
- index privilege 518
- INDEX privilege 515
- index searching
 - details 266
- index statistics
 - viewing SQL or XQuery statement statistics 469
- index type
 - unique index 258

- indexes
 - CREATE INDEX statement 261
 - CREATE UNIQUE INDEX statement 261
 - creating
 - overview 256
 - creation 467
 - definition of 258
 - DROP INDEX statement 327
 - dropping 327
 - estimating space requirements 272
 - nonprimary 327
 - nonunique 261
 - online reorganization 258, 261
 - optimizing number 258
 - performance tips for 260
 - primary versus user-defined 258
 - privileges
 - description 518
 - renaming 326
 - selectivity 268
 - specifications and extensions 258
 - unique 261
 - uniqueness for primary key 223
 - user-defined extended index
 - type 265
 - viewing SQL or XQuery statement details 469
- indexes exploitation 267
- infopops
 - setting for DB2 administration tools 435
 - turning on 435
- Information Center
 - updating 697
 - versions 695
 - viewing in different languages 696
- informational constraints 233
- infraction information
 - viewing 415
- INSERT privilege 515
- inserting
 - effects of LBAC on 563
- instance level profile registry 65
- instance owner 36
- instance profile registry 65
- instance user
 - setting the environment 34
- instances
 - add 41
 - adding partition servers 144
 - altering 275
 - auto-starting 42
 - creating 34
 - UNIX 39
 - Windows 40
 - creating additional 38
 - default 34
 - default, setting 25
 - definition 34
 - directory 34
 - disadvantages 34
 - listing 41
 - listing database partition servers 143
 - multiple 16
 - multiple on UNIX 36
 - multiple on Windows 37

- instances (*continued*)
 - overview of 16
 - owner 36
 - partition servers
 - changing 145
 - dropping 147
 - quiescing 42
 - reasons for using 34
 - removing 278
 - running multiple 27
 - setting the current 67
 - starting on UNIX 4
 - starting on Windows 4
 - stopping on UNIX 13
 - stopping on Windows 14
 - unquiescing 42
 - updating the configuration
 - UNIX 276
 - Windows 277
- inter-partition query parallelism
 - enabling 7
- intra-partition parallelism
 - enabling 7

J

- Java virtual machine
 - setup on DAS 101
- JCC Type 4 drivers
 - with client reroute 54
- Journal
 - overview 418
- Journal File System (JFS) 159

K

- Kerberos
 - authentication details 496
 - authentication type 490
 - security protocols
 - third party authentication 490
- keys
 - foreign
 - changing 311
 - primary
 - changing 309
 - unique
 - adding or changing 312
 - changing 313
- Known discovery 107
- KRB_SERVER_ENCRYPT authentication
 - type 490

L

- label-based access control (LBAC)
 - inserting data protected by 563
 - overview 538
 - protecting data using 558
 - reading data protected by 560
 - security label comparisons 550
 - updating data protected by 565
- large object (LOB) data types
 - column considerations 221
- large page support
 - AIX 64-bit environment 12

- launchpads
 - using 385
- LBAC (label-based access control)
 - inserting data protected by 563
 - overview 538
 - protecting data using 558
 - reading data protected by 560
 - security label comparisons 550
 - updating data protected by 565
- LBAC credentials
 - description 538
- LBAC protected data
 - adding protection 558
 - description 538
- LBAC protected tables
 - description 538
- LBAC rule exemptions
 - description and use 556
 - effect on security label comparisons 550
- LBAC rule sets
 - DB2LBACRULES 552
 - description 551
 - use in comparing security labels 550
- LBAC security administrator
 - description 538
- LBAC security label components
 - effect on security label comparisons 550
- LBAC security labels
 - ARRAY component type 543
 - compatible data types 547
 - components 541
 - description 538
 - how compared 550
 - SET component type 543
 - string format 549
 - TREE component type 544
 - use 547
- LBAC security policies
 - adding to a table 558
 - description 538
 - description and use 540
- LDAP (Lightweight Directory Access Protocol)
 - attaching remotely 583
 - cataloging a node entry 581
 - configuring DB2 576
 - creating a user 577
 - DB2 Connect 589
 - deregistering
 - databases 584
 - servers 582
 - description 573
 - directory service 181
 - disabling 589
 - enabling 588
 - extending directory schema 591
 - object classes and attributes 598
 - refreshing entries 584
 - registering
 - databases 582
 - DB2 servers 578
 - host databases 586
 - searching
 - directory domains 585
 - directory partitions 585

- LDAP (Lightweight Directory Access Protocol) (*continued*)
 - security 589
 - setting registry variables 587
 - supporting 575
 - updating protocol information 580
 - Windows 2000 active directory 591
- LDAP clients
 - rerouting 580
- legend
 - Control Center 380
- length limits
 - source data types 250
- LEVEL2 PCTFREE clause 261
- License Center
 - definition 411
 - managing licenses 64
 - overview 411
 - viewing user details 415
- license policies
 - viewing 414
- licenses
 - adding 412
 - changing 413
 - removing 416
- licensing information
 - viewing 413
- lightweight directory access protocol (LDAP)
 - attaching remotely 583
 - cataloging a node entry 581
 - configuring DB2 576
 - creating a user 577
 - DB2 Connect 589
 - deregistering
 - databases 584
 - servers 582
 - description 573
 - directory service 181
 - disabling 589
 - enabling 588
 - extending directory schema 591
 - object classes and attributes 598
 - refreshing entries 584
 - registering
 - databases 582
 - DB2 servers 578
 - host databases 586
 - searching
 - directory domains 585
 - directory partitions 585
 - security 589
 - setting registry variables 587
 - supporting 575
 - updating protocol information 580
 - Windows 2000 active directory 591
- LOAD database authority 511
- LOAD privilege 511
- Load wizard
 - loading data into a table 237
- loading data
 - enabling parallelism 10
 - into a table
 - using a Load wizard 237
- LOB (large object) data types
 - column considerations 221

- local database directory
 - description 178
 - viewing 179
- local system account 485
- LOCK TABLE statement
 - when using CREATE INDEX 261
- log files
 - administration 182
- logging
 - raw devices 163
- logical nodes; see database partition servers 30, 138
- logs
 - audit 621
 - Policy Evaluation 441
- LONGDATACOMPAT
 - retrieving access plan 468

M

- machine list
 - for partitioned database environment 137
- materialized query tables
 - behavior 206
 - with partitioned tables 206
- materialized query tables (MQTs)
 - altering properties 335
 - creating 201
 - dropping 365
 - populating 205
 - refreshing data 336
 - user-maintained 204, 205
- maxRetriesForClientReroute 45
- menus
 - changing fonts 437
 - DB2 Help 375
 - DB2 Tools 374
- MERGE statement
 - updating table and view contents 360
- messages
 - audit facility 636
 - default notification, changing 423
 - viewing in Journal 418
- method privileges 518
- MINPCTUSED clause 261
- modifying a table 295
- monitoring
 - rah processes 134
- MQTs (materialized query tables)
 - altering properties 335
 - creating 201
 - dropping 365
 - populating 205
 - refreshing data 336
 - user-maintained 204, 205
- multiple DB2 copies
 - roadmap 15
 - setting the default instance 25
- multiple instances 16
 - UNIX 36
 - Windows 37
- multiple logical nodes
 - configuring 31

N

- naming conventions
 - restrictions
 - general 663
 - Windows 678
- naming rules
 - DB2 objects 663
 - delimited identifiers and object names 665
 - federated database objects 666
 - general 663
 - national languages 668
 - objects and users 489
 - restrictions 663
 - schema names 667
 - Unicode 669
 - users, user IDs and groups 666
 - workstations 667
- Netscape
 - LDAP directory support 593
- NEXTVAL expression 234
- nicknames
 - privileges
 - indirect through packages 524
- NO FILE SYSTEM CACHING
 - clause 159
- node configuration files
 - creating 81
- node directories 179
- node level profile registry 65
- nodegroups (database partition groups)
 - creating 115
- non-buffered I/O
 - enabling on UNIX 159
- nonprimary indexes
 - dropping 327
- notices 701
- notification message
 - default 423
- notifications
 - changing the default message 423
 - enabling or disabling
 - using the Health Center Status Beacon 448
 - viewing in Journal 418
- null column definition 217

O

- object tree
 - adding databases 390
 - adding IMSplexes 390
 - adding instances 390
 - adding systems 390
 - adding z/OS subsystems 389
 - expanding and collapsing 389
 - refreshing objects 391
- objects
 - modifying
 - statement dependencies 366
 - performance on Windows 687
 - schemas for grouping 6
- objects in custom folders
 - deleting 391
- operand
 - definition 457

- operators
 - definition 457
 - viewing SQL or XQuery statement details 469
- optimizer
 - definition 458
- ordered domain list
 - authentication using 682
- ordering DB2 books 694
- overviews
 - DB2 UDB for z/OS health monitor 441
- ownership
 - database objects 501, 609

P

- packages
 - access privileges with queries 523
 - definition 459
 - dropping 327
 - inoperative 366
 - invalid
 - after adding foreign key 309
 - dependent on dropped indexes 327
 - owner 523
 - privileges 517
 - revoking privileges 521
 - viewing explainable statements 474
- page fetch pairs
 - viewing SQL or XQuery statement statistics 469
- PAGE SPLIT clause 261
- parallelism
 - intra-partition
 - enabling 7
- partitioned database environments
 - duplicate machine entries, eliminating 138
 - specifying machine list 137
- partitioned databases
 - errors when adding nodes 128
- partitioned tables
 - altering 210, 336, 338, 356, 358
 - attaching 346
 - avoiding a mismatch 348
 - converting 348
 - creating 193, 198
 - data rotating 339
 - detaching 352
 - loading 198
 - migrating 348
 - moving data 342
 - restrictions 338
 - rolling in data 342
 - rolling out data 342
 - with materialized query tables 206
- partitioning data
 - administration 9
- partitioning keys
 - adding or changing for a table 297
- partitions
 - dropping 125
- partitions, data
 - add 336
 - adding 356
- partitions, data (*continued*)
 - attach, rolling-in data 336
 - attached 338
 - attaching 210, 339
 - creating 195
 - defining the range 195
 - detach, rolling-out data 336
 - detached 338, 354
 - detaching 339
 - dropping 358
 - rolling in data, attaching 346
 - rolling in, attaching 342
 - rolling out, detaching 342
- passwords
 - maintaining
 - on servers 667
- performance
 - accessing remote information 688
 - catalog information, reducing contention for 9
 - displaying information 687
 - enable remote access to information 686
 - materialized query table 201
 - resetting values 688
 - Windows 687
- Performance Configuration wizard
 - invoking 83
 - renamed to Configuration Advisor 279
- performance monitor
 - Windows 685
- permissions
 - column-specific protection 538
 - row-specific protection 538
- plug-ins
 - adding toolbar buttons 398
 - architecture 395
 - basic menu action separators 401
 - basic menu actions 399
 - compiling 396
 - developing 397
 - guidelines 395
 - menu items, restricting display 402
 - positioning menu items 400
 - running 396
 - setting tree object attributes 406
- policies
 - changing 413
- policy evaluation log
 - DB2 UDB for z/OS health monitor 441
- populating a typed table 217
- port numbers
 - range
 - defining 144
- pre-filtering
 - objects in the Control Center 394
- PRECOMPILE command
 - OWNER option 523
- predicate
 - definition 459
- prefetch size
 - enabling automatic adjustment 288
- prefix
 - sequences 135
- PREVVAL 234
- primary keys
 - add to table 309
 - adding or changing 297
 - changing 309
 - constraints 223
 - DROP PRIMARY KEY clause, ALTER TABLE statement 316
 - dropping
 - using the Control Center 316
 - primary index 223, 258
 - privileges required to drop 316
 - when to create 223
- printed books
 - ordering 694
- privileges
 - ALTER 515
 - CONTROL 515
 - create view for information 613
 - DELETE 515
 - description 501
 - EXECUTE 518
 - GRANT statement 519
 - granting to new groups 530
 - granting to new users 534
 - hierarchy 501
 - implicit for packages 501
 - INDEX
 - description 515, 518
 - indirect 524
 - individual 501
 - INSERT 515
 - ownership (CONTROL) 501
 - package
 - creating 517
 - REFERENCES 515
 - retrieving
 - authorization names with 610
 - for names 613
 - REVOKE statement 521
 - schema 514
 - SELECT 515
 - SETSESSIONUSER 513
 - system catalog listing 609
 - table 515
 - table space 515
 - tasks and required authorities 608
 - UPDATE 515
 - USAGE 518
 - view 515
- problem determination
 - online information 699
 - tutorials 699
- procedure privileges 518
- profile registry 65
- properties of columns
 - changing 304
- protected data (LBAC)
 - adding protection 558
- protecting data with LBAC 558
- PUBLIC clause
 - database authorities, figure 511
- purging
 - task history records
 - Task Center 416

Q

- qualified object names 6
- queries
 - rewrite, materialized query table 201
- query optimization class
 - definition 460
- QUIESCE_CONNECT database
 - authority 511
- quiescing
 - databases 186
 - instances 42
 - tables 239

R

- rah command
 - controlling 139
 - description 131
 - determining problems 141
 - environment variables 139
 - introduction 130
 - monitoring processes 134
 - overview 130
 - prefix sequences 135
- RAHCHECKBUF environment
 - variable 133
- RAHDOTFILES environment
 - variable 140
- RAHOSTFILE environment
 - variable 137
- RAHOSTLIST environment
 - variable 137
- RAHWAITTIME environment
 - variable 134
- recursively invoked 135
- running commands in parallel 133
- setting the default environment
 - profile 141
 - specifying
 - as a parameter or response 132
 - database partition server list 137
- RAHCHECKBUF environment
 - variable 133
- RAHDOTFILES environment
 - variable 140
- RAHOSTFILE environment variable 137
- RAHOSTLIST environment variable 137
- RAHTREETHRESH environment
 - variable 135
- RAHWAITTIME environment
 - variable 134
- range-clustered tables
 - access path determination 215
 - examples 213
 - guidelines 216
- ranges
 - defining for data partitions 195
 - generating 195
 - restrictions 195
- raw devices 149
- raw I/O
 - setting up on Linux 164
 - specifying 163
- raw logs 163
- rebalancing data across containers 285

- recommended actions
 - viewing, submitting, saving 443
- records
 - audit 621
- recovery
 - allocating log during database creation 182
 - summary tables, inoperative 361
 - views, inoperative 331
- redistributing data 128
 - across database partitions 281
- referenced column groups
 - viewing SQL or XQuery statement details 469
- referenced columns
 - viewing SQL or XQuery statement details 469
- REFERENCES clause
 - delete rules 227
 - use of 227
- REFERENCES privilege 515
- referential constraints
 - defining 224
 - PRIMARY KEY clause, CREATE/ALTER TABLE statements 224
 - REFERENCES clause, CREATE/ALTER TABLE statements 224
- refreshing
 - data in materialized query table 336
 - DB2 UDB for z/OS health monitor 442
 - objects in the object tree 391
- registry variables
 - aggregate 75
 - DB2_CONNRETRIES_INTERVAL 52
 - db2_connretries_interval 45
 - DB2_CONNRETRIES_INTERVAL 70
 - DB2_INDEX_TYPE2 70
 - DB2_LIC_STAT_SIZE 70
 - DB2_MAX_CLIENT_CONNRETRIES 52, 70
 - db2_max_client_connretries 45
 - DB2_VIEW_REOPT_VALUES 70
 - DB2ACCOUNT 70
 - DB2BIDI 70
 - DB2CODEPAGE 70
 - DB2DBDFT 70
 - DB2DBMSADDR 70
 - DB2DISCOVERYTIME 70
 - DB2GRAPHICUNICODESERVER 70
 - DB2INCLUDE 70
 - DB2INSTDEF 70
 - DB2INSTOWNER 70
 - DB2LOCALE 70
 - DB2NBDISCOVERRCVBUFS 70
 - DB2SLOGON 70
 - DB2TERRITORY 70
 - DB2TRACEFLUSH 70
 - DB2TRACENAME 70
 - DB2TRACEON 70
 - DB2TRCSYSERR 70
 - DB2YIELD 70
 - declaring 68
 - environment variables 65

- related objects
 - showing
 - in the Control Center 270
 - validating 271
- remote
 - administration 104
 - performance 688
- removing
 - columns 320
- renaming
 - indexes 326
 - table spaces 290
 - tables 326
- REORG-recommended alter 300
- reorganization utility
 - binding to a database 183
- rerouting clients 44
 - LDAP 580
- resizing
 - table space 154
- restore database
 - implications 59
- Restore wizard
 - restoring data 388
- restoring
 - data
 - using the Restore wizard 388
 - databases, enabling I/O
 - parallelism 11
 - table spaces, enabling I/O
 - parallelism 11
- RESTRICT semantic
 - for DROP COLUMN 300
- restrictions
 - automatic storage 62
 - naming
 - Windows 678
- RESTRICTIVE option, CREATE DATABASE 613
- retryIntervalForClientReroute 45
- REVOKE statement
 - example 521
 - implicit issuance 522
 - use 521
- revoking
 - LBAC security labels 547
- roadmaps
 - automatic client reroute 45
 - multiple DB2 copies 15
- row blocking
 - see cursor blocking 460
- row compression 188
 - definition 187
- rows
 - deleting LBAC protected 569
 - effect of LBAC on reading 560
 - inserting LBAC protected 563
 - protecting a row with LBAC 558
 - updating LBAC protected 565
- rule sets (LBAC)
 - description 551
 - exemptions 556
- running tasks
 - immediately 421
- runstats
 - using 468

S

- saved schedules
 - managing 429
- scalar functions
 - creating 243
- scenarios
 - defining an index extension 268
- schedule settings
 - enabling 419
- scheduler 420
 - concept 420
 - DB2 administration server (DAS) 96
- scheduler system 425
- schedules
 - managing 429
- scheduling
 - tasks 422
- scheduling scheme
 - setting, default 449
- schema names
 - description 667
- schemas
 - copying schemas and objects 170
 - creating 168
 - description 6
 - dropping 294
 - restarting failed copy schema
 - operation 173
 - SESSION 364
 - setting 169
- scope
 - adding 305
- SEARCH discovery
 - in discovery parameter of Known Discovery 107
- SECADM database authority 501, 508, 511
- SECLABEL
 - description 557
- SECLABEL_BY_NAME
 - description 557
- SECLABEL_TO_CHAR
 - description 557
- security
 - CLIENT level 490
 - column-specific 538
 - db2extsec command
 - using 486
 - disabling extended security 486
 - enabling extended security 486
 - extended security 486
 - label-based access control (LBAC) 538
 - maintaining passwords
 - on servers 667
 - planning for 481
 - risks 616
 - row-specific 538
 - UNIX considerations 489
 - Windows 486
 - description 675
 - domain security 683
 - services 680
 - users 485
- security administrator (SECADM) database authority 501, 508, 511
- security labels (LBAC)
 - ARRAY component type 543
 - compatible data types 547
 - components 541
 - SET component type 543
 - string format 549
 - TREE component type 544
 - use 547
- security policies (LBAC)
 - description and use 540
- SELECT clause
 - used in a view 251
- SELECT privilege 515
- selectivity of predicates
 - definition 461
- sequences 461
 - altering 319
 - comparing with IDENTITY columns 235
 - creating 234
 - dropping 320
 - privileges 518
- server administration tools
 - setting startup property 434
- SERVER authentication type 490
- SERVER_ENCRYPT authentication type 490
- servers
 - alternate 45, 49
- service-level information
 - for administration tools 370
- SET ENCRYPTION PASSWORD statement 527
- SET INTEGRITY statement
 - checking for constraint violations 230
- SETSESSIONUSER privilege 513
- settings
 - default environment profile for rah 141
 - IMS options 450
 - schema 169
- shutting down
 - administration tools 369
- SIGTTIN message 132
- SMS (system managed space)
 - table spaces
 - adding containers 289
 - creating 149
- source data types 249
 - length limits 250
- source tables
 - creating 210
- space compression
 - existing tables 295
 - new tables 187
 - tables 187
- space requirements
 - for tables and indexes
 - estimating 272
- sparse file allocation 221
- SQL (Structured Query Language)
 - keywords 665
- SQL or XQuery statement
 - viewing text 469
- SQL statements
 - displaying help 695
 - dynamically explaining 464
- SQL statements (*continued*)
 - explained
 - viewing history 476
 - inoperative 366
- staging tables
 - creating 211
 - deleting contents of 362
 - dropping 365
- star joins
 - definition 462
- START MVS system command 441
- starting
 - DB2
 - UNIX 4
 - Windows 4
 - DB2 UDB for z/OS health monitor 442
- startup options
 - setting 436
- startup property
 - setting 434
- static SQL or XQuery statements
 - definition 463
 - EXECUTE privilege for database access 523
- stdin 132
- STOP MVS system command 441
- stopping
 - DB2
 - UNIX 13
 - Windows 14
 - DB2 UDB for z/OS health monitor 442
- storage
 - automatic, for databases 54
 - automatic, for table spaces 58
- storage paths
 - automatic
 - adding 64
- stored procedures
 - altering a table 324
- stripe sets 285
- structured types
 - altering 333
- submenus
 - creating 401
- success code sets 420
 - concept 420
 - managing 430
- summary tables
 - recovering inoperative 361
- Sun One Directory Server
 - extending directory schema for 596
- SWITCH ONLINE clause 291
- synonyms
 - DB2 for OS/390 or z/Series 254
- SYSCAT catalog views
 - for security issues 609
- SYSCATSPACE table spaces 148
- system administration (SYSADM)
 - authority
 - description 506
 - privileges 506
 - system catalog tables
 - description 175

- system catalogs
 - dropping
 - tables 363
 - view implications 330
 - privileges listing 609
 - retrieving
 - authorization names with
 - privileges 610
 - names with DBADM
 - authority 611
 - names with table access
 - authority 612
 - privileges granted to names 613
 - security 613
 - system control authority (SYSCTRL) 507
 - system database directory
 - overview 178
 - viewing 179
 - system maintenance authority (SYSMAINT) 508
 - system monitor authority (SYSMON) 510
 - system names
 - changing 383
 - system temporary table spaces 158

T

- table
 - altering 295
- table function statistics
 - viewing SQL or XQuery statement
 - statistics 469
- table information
 - displaying in the Control Center
 - contents pane 392
- table objects
 - altering 295
 - creating 187
- table partitions
 - managing 297
- table properties
 - changing 299
- table spaces
 - adding
 - containers 285
 - automatic resizing 154
 - automatic storage 58
 - automatic storage, regular and
 - large 63
 - changing 284
 - containers
 - extending 286
 - file example 149
 - file system example 149
 - creating
 - description 149
 - in database partition groups 163
 - definition 148
 - device container example 149
 - dropped table recovery 149
 - dropping
 - system temporary 292
 - user 291
 - user temporary 293
 - enabling I/O parallelism 11
 - initial 148

- table spaces (*continued*)
 - privileges 515
 - renaming 290
 - resizing container 286
 - separating types of data,
 - example 190
 - switching states 291
 - system temporary 158
 - temporary automatic storage 57
 - user temporary 159
 - viewing SQL or XQuery statement
 - details 469
 - without file system caching 159
- table statistics
 - viewing SQL or XQuery statement
 - statistics 469
- table user-defined functions (UDFs)
 - description 243
- tables
 - add referential constraints 309, 310
 - adding
 - columns, new 304
 - ALTER TABLE statement 304
 - ALTERING partitioned tables 356, 358
 - altering using stored procedures 324
 - changing
 - distribution keys 318
 - changing attributes 298
 - converting 198
 - copying 296
 - CREATE TABLE statement 217
 - creating 187
 - in partitioned databases 191
 - creation
 - overview 189
 - defining
 - check constraints 228
 - dimensions 235
 - referential constraints 224
 - unique constraints 223
 - dropping 363
 - effect of LBAC on reading 560
 - estimating space requirements 272
 - explain
 - creating 466
 - generated columns 219, 321
 - identity columns 220
 - inserting into LBAC protected 563
 - loading data using the Load
 - wizard 237
 - making fully accessible 238
 - materialized query tables 206
 - migrating into partitioned tables 198
 - mismatch 210
 - naming 217
 - partitioned tables 206
 - protecting with LBAC 538, 558
 - quiescing 239
 - range-clustered 216
 - removing
 - rows 307
 - renaming 326
 - retrieving names with access to 612
 - revoking privileges 521
 - source 210
 - staging, deleting contents of 362
 - target 210

- tables (*continued*)
 - tips for adding constraints 309, 310
 - updating using MERGE
 - statement 360
 - volatile, declaring 323
- task categories
 - managing 431
- Task Center
 - creating tasks 425
 - description 416
 - editing tasks 425
 - enabling scheduling settings 419
 - overview 416
 - purging task history records 416
- tasks
 - authorizations 608
 - creating or editing 425
 - creating or editing 425
 - running immediately 421
 - running now 421
 - scheduling 420, 422
 - viewing in Journal 418
- TCP_KEEPALIVE
 - operating system configuration
 - parameter 53
- temporary tables
 - dropping a user-defined 364
 - user-defined 212
- TEMPSPACE1 table space 148
- termination character
 - setting for command statement 434
- terms and conditions
 - use of publications 699
- toolbars
 - primary 371
 - secondary 373
- tools
 - catalog database 96
- Tools Settings
 - overview 432
- trace facility 393
- triggers
 - benefits 240
 - creating 240
 - dependencies 242
 - dropping 329
 - updates
 - update view contents 328
- troubleshooting
 - online information 699
 - tutorials 699
- trust relationships 679
- trusted clients
 - CLIENT level security 490
- tutorials
 - troubleshooting and problem
 - determination 699
 - Visual Explain 698
- type mapping
 - creating 248
 - dropping 334
- typed tables
 - creating 216
 - deleting rows 362
 - populating 217
 - updating rows 362

U

- unavailable status
 - of databases 391
- Unicode (UCS-2)
 - identifiers 669
 - naming rules 669
- uninstalling
 - DB2 Copies 28
- union all views
 - converting 198
- unique constraints
 - adding 313
 - defining 223
 - dropping 315
- unique key values
 - generating
 - using sequences 461
- unique keys
 - adding or changing 312
 - changing 313
- UNIX File System (UFS)
 - supported configuration 159
- unquiescing
 - databases 186
 - instances 42
- UPDATE privilege 515
- updates
 - DAS configuration 110
 - DB2 Information Center 697
 - Information Center 697
 - typed table 362
- updating
 - effects of LBAC on 565
 - view contents using triggers 328
- USAGE privilege 518
- user authentication
 - Windows 678
- user IDs
 - naming rules 666
 - selecting 481
- user table spaces 291
- user temporary table spaces
 - creating 159
 - dropping 293
- user-defined extended index types 265
- user-defined functions (UDFs)
 - creating 243
 - database authority to create
 - non-fenced 511
 - dropping 333
 - types 243
 - viewing SQL or XQuery statement
 - details 469
- user-defined temporary tables
 - creating 212
 - dropping 364
- user-defined types (UDTs)
 - creating 246
 - distinct types
 - creating 247
 - dropping 334
 - structured types 248
- users
 - selecting new tasks for 427
- USERSPACE1 table space 148
- utility execution options
 - setting, for z/OS 437

- utility operations
 - constraint implications 227

V

- validating
 - related objects 271
- value compression
 - definition 187
 - existing tables 187, 295
- VARCHAR data type
 - in table columns 305
- VERITAS Storage Foundation 159
- views
 - access control to table 525
 - access privileges, examples of 525
 - altering 330
 - column access 525
 - creating 251
 - data integrity 251
 - data security 251
 - dropping 330
 - dropping implications for system
 - catalogs 330
 - for privileges information 613
 - inoperative 331
 - recovering inoperative 331
 - removing rows 307
 - restrictions 330
 - row access 525
 - triggers to update 328
 - updating using MERGE statement 360
- Visual Explain
 - access plan graph 453
 - access plans 452
 - definition 463
 - overview 451
 - tutorial 698
 - up-level and down-level support 478
- vmo
 - AIX system command 12
- vmtune
 - AIX system command 12

W

- Windows
 - active directory, DB2 objects
 - configuring on Windows 593
 - active directory, object classes and attributes
 - configuring on Windows 598
 - adding database partitions 122
 - extending the directory schema
 - Windows 2000 591
 - Performance Monitor 685
 - security 486
- Windows Management Instrumentation (WMI)
 - DB2 database integration 672
 - description 671
- Windows scenarios
 - client authentication
 - Windows client 677
 - server authentication 676

- Windows support
 - local system account (LSA) 485
- Windows user group
 - access token 483
- wizards
 - Add Database Partitions wizard 124
 - Create Table 190
 - Performance Configuration 279
 - using 385
- workstations
 - (nname), naming rules 667
- write-down
 - description 552
- write-up
 - description 552

X

- XQuery statements
 - dynamically explaining 464
 - explained
 - viewing history 476
 - inoperative 366

Z

- z/OS subsystems
 - adding to object tree 389

Contacting IBM

To contact IBM in your country or region, check the IBM Directory of Worldwide Contacts at <http://www.ibm.com/planetwide>

To learn more about DB2 products, go to <http://www.ibm.com/software/data/db2/>.



Printed in USA

SC10-4221-00



Spine information:

IBM DB2 DB2 Version 9

Administration Guide: Implementation

