CHAPTER 3

Access and Security

In This Chapter

- ✔ Subsystem access
- ✔ Data set protection
- ✔ Authorization IDs
- ✔ Trusted context and roles
- ✔ Authorities and privileges
- ✔ Auditing

Whenever you store data in a relational database management system, security is an important consideration. In this chapter, we discuss controlling data access using many different methods. Access to data within DB2 is controlled at several levels, including the subsystem, database object, and application plan/package. We discuss user ID and password authentication and describe how to configure groups of typical database users, such as database administrators, system administrators, transactional processing personnel, and decision support users. Each of these user types may require different access privileges. As a final piece to our security discussion, we explain how to audit access to DB2 objects so you can monitor access to and manipulation of data.

As Figure 3.1 depicts, there are several routes from a process to DB2 data, with controls on every route.

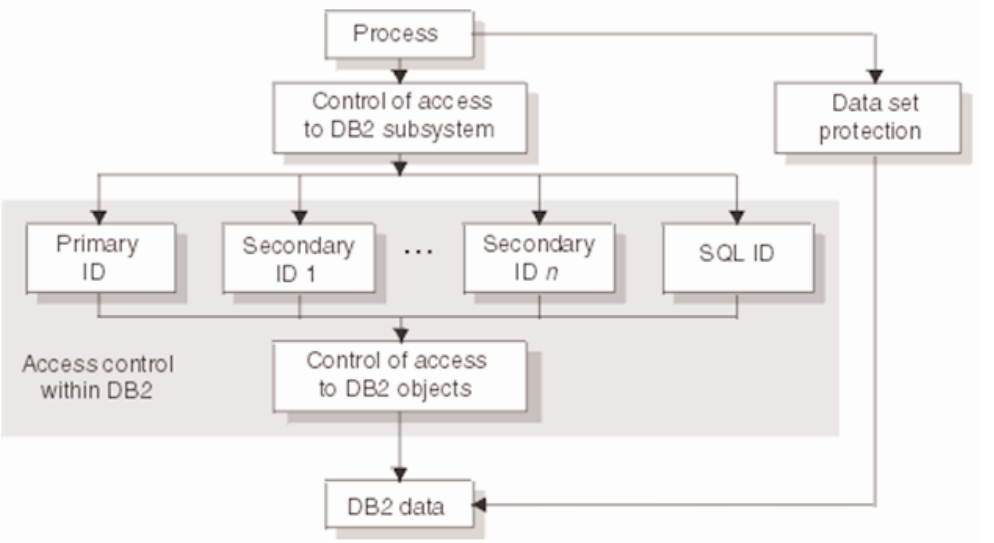

Figure 3.1: DB2 data access control

With each new release, DB2 gets bigger, faster, and more secure. Over the years, it has recognized and addressed the following security problems:

- Privilege theft or mismanagement
- Application or application server tampering
- Data or log tampering
- Storage media theft
- Unauthorized object access

To address these areas, DB2 offers the following security solutions:

- Authentication
- Authorization
- Data integrity

- Confidentiality
- system integrity
- Audit

DB2 Subsystem Access

You can control whether a process can gain access to a specific DB2 subsystem from outside DB2. A common procedure is to grant access only through the Resource Access Control Facility or a similar security system. With this approach, you define profiles for access to DB2 from various environments (and DB2 address spaces) as resources to RACF. You identify these profiles by specifying the subsystem and the environment. Environments include the following:

Environment	Description
MASS	For IMS
SASS	For CICS
DIST	For DDF
RRSAF	For RRSAF
BATCH	For TSO, CAF, utilities

Each request to access DB2 is associated with an identifier. When a request is made, RACF verifies whether this ID is authorized for DB2 resources and either permits or does not permit access to DB2.

The RACF system provides several advantages of its own. For example, it can

- identify and verify the ID associated with a process
- connect those IDs to RACF group names
- log and report unauthorized attempts to access protected resources

The RACF resource class for DB2 is DSNR, and this class is contained in the RACF descriptor table. To control access, you define a profile name as a member of class DSNR for every combination of subsystem and environment you'll be using. You can then issue commands to give authority to groups that are authorized for class DSNR.

For example, the following PERMIT command lets users run batch jobs and utilities on a subsystem.

```
PERMIT DSN.BATCH CLASS(DSNR) ID(DB2USER) ACCESS(READ)
```

You can also use PERMIT to take away DB2 access from a user:

```
PERMIT DSNP.TSO CLASS(DSNR) ID(DB2USER) ACCESS(NONE)
```

Authorization Control with Exit Routines

You can also control access to DB2 subsystems through exit routines. DB2 provides two exit points for authorization routines, one in connection processing and one in sign-on processing. Both are important for ID assignment. You need a routine for each exit, and IBM supplies default routines for each type: DSN3@ATH for connections and DSN3@SGN for sign-ons.

DB2 provides a third exit point (DSNX@XAC) that lets you furnish your own access-control routines or use RACF (or the equivalent) to perform system authorization checking. When DB2 invokes an authorization routine, it passes three possible functions to it:

- Initialization (DB2 startup)
- Authorization check
- Termination (DB2 shutdown)

The exit routine may not be called in the following situations:

- If the user is Install SYSADM or Install SYSOPR
- If DB2 security has been disabled (i.e., if you specified NO for the USE PROTECTION installation field on the DSNTIPP panel)
- If a prior invocation of the routine indicated the routine should not be called again
- If a GRANT statement is being executed

Local DB2 Access

Even before reaching DB2, a local DB2 user is subject to several checks. For example, if you're running DB2 under TSO and using the TSO logon ID as the DB2 primary authorization ID, that ID is verified with a password when the user logs on. Once the user gains access to DB2, a user-written or IBM-supplied exit routine connected to DB2 can check the authorization ID further, change it, and associate it with secondary IDs (which we discuss later). In providing these functions, DB2 can use the services of an external security system.

Remote Access

Remote users, too, are subject to several checks before reaching your DB2. You can use RACF or a similar security subsystem. RACF can

- verify an identifier associated with a remote attachment request and check it with a password.
- generate PassTickets on the sending side. Used instead of a password, a PassTicket lets a user gain access to a host system without sending the RACF password across the network.

DB2's communications database does permit some control of authentication in that you can cause IDs to be translated before sending them to the remote system. For more information about accessing DB2 and the CDB, see Chapter 2.

IMS and CICS Security

You can also control DB2 access from within IMS or CICS.

IMS terminal security lets you limit the entry of a transaction code to a particular logical terminal (LTERM) or group of LTERMs in the system. To protect a particular program, you can authorize a transaction code to be entered only from any terminal on a list of LTERMs. As an alternative, you can associate each LTERM with a list of the transaction codes a user can enter from that LTERM. IMS then passes the validated LTERM name to DB2 as the initial primary authorization ID.

CICS transaction code security works with RACF to control the transactions and programs that can access DB2. Within DB2, you can use the ENABLE and DISABLE options of the bind operation to limit access to specific CICS subsystems.

Kerberos Security

Kerberos security is a network security technology developed at the Massachusetts Institute of Technology. DB2 for z/OS can use Kerberos security services to authenticate remote users. With Kerberos security services, remote end users access DB2 when they issue their Kerberos name and password. This same name and password is used for access throughout the network, so a separate z/OS password to access DB2 isn't necessary.

Kerberos security technology doesn't require passwords to flow in readable text, making it secure even in client/server environments. This flexibility is possible because Kerberos employs an authentication technology that uses encrypted tickets that contain authentication information for the end user.

DB2 support for Kerberos security requires the z/OS SecureWay Security Server Network Authentication and Privacy Service and the z/OS SecureWay Security Server (formerly known as RACF), or the functional equivalent. The Network Authentication and Privacy Service provides Kerberos support and relies on a security product (e.g., RACF) to provide registry support. The SecureWay Security Server enables administrators already familiar with RACF commands and RACF ISPF panels to define Kerberos configuration and principal information.

You can use Kerberos security only if you have the z/OS SecureWay Security Server.

Secure Sockets Layer Support

DB2 exploits the z/OS Application Transparent–Transport Layer Security (AT-TLS) function in the TCP/IP stack to provide TLS for DB2 clients that require secure connections. AT-TLS performs TLS on behalf of the application by invoking the z/OS system SSL in the TCP transport layer of the stack. DB2's SSL support

provides protected connections between DB2 servers. With SSL support, a DB2 server can optionally listen on a secondary secure port for inbound SSL connections. Similarly, a DB2 requester can optionally send encrypted data across the network through an SSL connection to the server.

Protection Against Denial-of-Service Attacks

In a denial-of-service attack, an attacker tries to prevent legitimate users from accessing information or services. By targeting a DB2 server and its network connection, an attacker might be able to prevent you from accessing data or other services that the server provides. The DB2 server guards against such attacks and provides a more secure operating environment for legitimate users.

Data-Set Protection

The data in a DB2 subsystem is contained in data sets. It's possible to access these data sets without going through DB2 at all. If the data is sensitive, you want to control that route.

If you're using the z/OS SecureWay Security Server (or a similar security system) to control access to DB2, the simplest way to control data-set access outside DB2 is to use RACF for that purpose, too. That means defining RACF profiles for data sets and permitting access to them for certain DB2 IDs.

If the data is very sensitive, you may want to consider encrypting it to protect against unauthorized access to data sets and backup copies outside DB2. You can use DB2 edit procedures or field procedures to encrypt data, and those routines can use the Integrated Cryptographic Service Facility (ICSF) of z/OS. Note that data compression is not a substitute for encryption. In some cases, the compression method doesn't actually shorten the data, and the data is then left uncompressed and readable. If you both encrypt and compress data, be sure to compress it first to obtain the maximum compression; then encrypt the result. When retrieving data, take the steps in reverse order: decrypt the data first, and then decompress the result.

DB2 Object Access

An individual process can be represented by a primary authorization ID, possibly one or more secondary IDs, and an SQL ID. The security and network systems and the DB2 connections that are made all affect the use of IDs.

DB2 controls access to objects by assigning privileges and authorities to either primary or secondary IDs. Object ownership also carries with it a set of related privileges over the object. An ID can own an object it creates, or it can create an object to be owned by another ID. Separate controls govern creation and ownership.

Executing a plan or package exercises implicitly all the privileges that the owner needed when binding it. Hence, granting the privilege to execute can provide a finely detailed set of privileges and can eliminate the need to grant other privileges separately.

You can use RACF access control to supplement or replace the DB2 GRANT and REVOKE statements.

In this section, we look at how privileges, authorities, and ownership work together to provide security for access to DB2 objects.

Authorization IDs

Every process that connects to or signs on to DB2 is represented by a set of one or more DB2 identifiers called *authorization IDs*. Authorization IDs can be assigned to a process by default procedures or by user-written exit routines.

Primary Authorization ID

A *primary authorization ID* is assigned to every process. Each process has only one primary authorization ID, and it is the ID that is normally used to uniquely identify the process.

Secondary Authorization ID

A *secondary authorization ID*, which can hold additional privileges, is optional. Secondary authorizations are often used for groups, such as RACF groups. A primary authorization ID can be associated with multiple secondary authorization IDs.

When you add a new user to an RACF group, that user is visible the next time he or she logs on to TSO.

Role

A role is available within a trusted context. You can define a role and assign it to an authorization ID in a trusted context. When associated with a role and using the trusted connection, the authorization ID inherits all the privileges granted to that role. We discuss roles and trusted contexts in more detail later.

Current SQL ID

Either the primary ID or the secondary ID can be the current SQL ID at any given time. Furthermore, one ID (either primary or secondary) is designated as the current SQL ID. You can change the value of the SQL ID during your session. For example, if DB2EXPT is your primary or one of your secondary authorization IDs, you can make it your current SQL ID by issuing the SQL statement

```
SET CURRENT SQLID ='DB2EXPT';
```

An ID with SYSADM authority (described later) can set the *current* SQL ID to any string of up to eight bytes, whether or not the ID is an authorization ID associated with the process that is running.

Trusted Contexts

DB2 9 helps you satisfy the need for data security and accountability by enabling you to create and use *trusted contexts* as another method to manage access to your DB2 servers. Within a trusted context, you can use trusted connections to reuse the authorization and switch users of the connection without the database server needing to authenticate the IDs.

A trusted context is an independent database entity that you can define based on a system authorization ID and connection trust attributes. The trust attributes specify a set of characteristics about a specific connection. These attributes include the IP address, domain name, or SERVAUTH security zone name of a remote client and the

job or task name of a local client. A trusted context lets you define a unique set of interactions between DB2 and the external entity, including the following abilities:

- The ability for the external entity to use an established database connection with a different user without the need to authenticate that user at the DB2 server. This support eliminates the need for the external entity to manage end-user passwords. Also, a database administrator can assume the identity of other users and perform actions on their behalf.
- The ability for a DB2 authorization ID to acquire one or more privileges within a trusted context that are not available to it outside that trusted context. You accomplish this by associating a role with the trusted context.

Several client applications support the trusted context:

- The DB2 Driver for JDBC and SQL introduces new APIs for establishing trusted connections and switching users of a trusted connection.
- The DB2 Driver for ODBC and CLI introduces new keywords for connecting APIs to establish trusted connections and switch users of a trusted connection.
- WebSphere Application Server 6.0 exploits the trusted context support through its "propagate client identity" property.

Trusted Connections

A *trusted connection* is a database connection that is established when the connection attributes match the attributes of a unique trusted context defined at the server. You can establish a trusted connection locally or at a remote location.

A trusted context establishes a trusted relationship between DB2 and an external entity, such as a middleware server. To determine whether a specific context can be trusted, DB2 evaluates a series of trust attributes. At this time, the only attribute DB2 considers is the database connection.

The relationship between a connection and a trusted context is established when the connection to the server is first created, and that relationship remains in place as long as that connection exists.

Roles

A *role* is a database entity, available only in a trusted context, that groups together one or more privileges and can be assigned to users. You can define a role and assign it to an authorization ID in a trusted context. When associated with a role and using the trusted connection, the authorization ID inherits all the privileges granted to that role.

A role can own database objects, a fact that helps eliminate the need for individual users to own and control database objects. A role owns objects if the objects are created in a trusted context with the role defined as the owner (by specifying the ROLE AS OBJECT OWNER clause in the trusted context definition). Databases, table spaces, tables, indexes, and views can be implemented in a trusted context with role as the owner of the created objects.

You can assign a role to an individual user or a group of users by defining a trusted context. A role thus offers a mechanism other than authorization IDs through which you can assign privileges and authorities. When you define a role for a trusted context, the role becomes the actual owner of the objects when you specify the ROLE AS OBJECT OWNER clause. As a result, roles give you the flexibility of authorization methods and help simplify the management of authentication.

If objects were created using a ROLE, you can remove the user ID to which the role was assigned without having to redo privileges or drop or re-create objects. For example, say a company creates an APP1 trusted context and an APP1_DBA role to limit exposure to an application and then assigns DBA1 to this role and all the objects. If DBA1 leaves the company and the ID is removed, the objects and privileges of the role remain untouched.

Defining Trusted Contexts

Before you can create a trusted connection, you must define a trusted context by specifying a system authorization ID and connection trust attributes.

A system authorization ID is the DB2 primary authorization ID used to establish the trusted connection. For local connections, the system authorization ID is derived as follows:

Source	System authorization ID
Started task (RRSAF)	USER parameter on JOB or RACF USER
TSO	TSO logon ID
BATCH	USER parameter on JOB

For remote connections, the system authorization ID is derived from the system user ID provided by an external entity, such as a middleware server.

Connection trust attributes identify a set of characteristics about the specific connection. These attributes are required for the connection to be considered a trusted connection. For a local connection, the connection trust attribute is the job or started task name. For a remote connection, the connection trust attribute is the client's IP address, domain name, or SERVAUTH security zone name. Table 3.1 describes the connection trust attributes.

Table 3.1: Connection trust attributes

Attribute	Description
ADDRESS	Specifies the client's IP address or domain name; used by the connection to communicate with DB2. The protocol must be TCP/IP.
SERVAUTH	Specifies the name of a resource in the RACF SERVAUTH class. This resource is the network access security zone name that contains the IP address of the connection to communicate with DB2.
ENCRYPTION	Specifies the minimum level of encryption of the data stream (network encryption) for the connection. **Value** — **Meaning** NONE — No encryption (the default) LOW — DRDA data stream encryption HIGH — SSL encryption
JOBNAME	Specifies the local z/OS started task or job name. The value of JOBNAME depends on the source of the address space. **Source** — **JOBNAME** Started task (RRSAF) — Job or started task name TSO — TSO logon ID BATCH — Job name on JOB statement

You cannot specify the JOBNAME attribute with the ADDRESS, SERVAUTH, or ENCRYPTION attribute.

Performing Tasks on Behalf of Others

If you have DBADM authority (described later), you can assume the identity of other users within a trusted context and perform tasks on their behalf. After successfully assuming the identity of a view owner, you inherit all the privileges from the ID that owns the view and can therefore perform the CREATE, DROP, and GRANT actions on the view.

To perform tasks on behalf of another user:

1. Define a trusted context. Make sure the SYSTEM AUTH ID is the primary authorization ID you use in SPUFI.
2. Specify the primary authorization ID as the JOBNAME for the trusted connection.
3. Specify the primary authorization ID of the user whose identity you want to assume.
4. Log on to TSO using your primary authorization ID.
5. Set the ASUSER option on the DB2I DEFAULTS panel to the primary authorization ID of the user whose identity you want to assume.
6. Perform the desired actions by using privileges of the specified user.

For example, let's assume you have DBADM authority (the minimum authority required), your primary authorization ID is DAN, and you want to drop a view owned by user SUSAN. You can issue the following statement to create and enable a trusted context called CTXLOCAL in which DAN can drop the selected view on SUSAN's behalf:

```
CREATE TRUSTED CONTEXT CTXLOCAL
BASED UPON CONNECTION USING SYSTEM AUTHID DAN
ATTRIBUTES (JOBNAME 'DAN')
ENABLE
ALLOW USE FOR SUSAN;
```

After logging on to TSO, set the ASUSER option to SUSAN in the DB2I DEFAULTS panel, and invoke SPUFI to process SQL statements. DB2 obtains the primary authorization ID DAN and JOBNAME DAN from the TSO log-on session, authenticates DAN, searches for the matching trusted context (CTXLOCAL), and establishes a trusted connection. DB2 then authenticates the primary authorization

ID (SUSAN) and validates all privileges assigned to SUSAN. After successful authentication and validation, you, DAN, can drop the view that is owned by SUSAN.

Explicit Privileges

You can grant several *explicit privileges* to a primary ID, secondary authorization ID, or role to grant that ID the privilege to perform a particular task. Certain granted privileges also provide an inherited authority (e.g., if you grant CREATEDBA to an ID, that ID will become DBADM over the database it creates). The privileges are grouped into several categories:

- Tables and views
- Plans
- Packages
- Collections
- Databases
- Subsystems
- Usage
- Schemas
- Distinct types or JARs
- Routines (functions or procedures)
- Sequences

Table 3.2 lists the available privileges that can be granted to a primary or secondary authorization ID, along with the type of usage associated with each privilege.

No specific authority exists for creating a view. To create a view, you must have the SELECT privilege from the table (or tables) on which the view is being created.

Additional privileges exist for statements, commands, and utility jobs.

Table 3.2: Explicit privileges	
Privilege	**Provides this usage**
	Table
ALTER	The ALTER TABLE statement, to change the table definition.
DELETE	The DELETE statement, to delete rows.
GRANT ALL	SQL statements of all table privileges.
INDEX	The CREATE INDEX statement, to create an index on the table.
INSERT	The INSERT statement, to insert rows.
REFERENCES	The ALTER or CREATE TABLE statement, to add or remove a referential constraint referring to the named table or to a list of columns in the table.
SELECT	The SELECT statement, to retrieve data from the table.
TRIGGER	The CREATE TRIGGER statement, to define a trigger on a table.
UPDATE	The UPDATE statement, to update all columns or a specific list of columns.
	Plan
BIND	The BIND, REBIND, and FREE PLAN subcommands, to bind or free the plan.
EXECUTE	The RUN command, to use the plan when running the application.
	Package
BIND	The BIND, REBIND, and FREE PACKAGE subcommands and the DROP PACKAGE statement, to bind or free the package and, depending on the installation option BIND NEW PACKAGE, to bind a new version of a package.
COPY	The COPY option of BIND PACKAGE, to copy a package.
EXECUTE	Inclusion of the package in the PKLIST option of BIND PLAN.
GRANT ALL	All package privileges.
	Collection
CREATE IN	Naming the collection in the BIND PACKAGE subcommand.
	Database
CREATETAB	The CREATE TABLE statement, to create tables in the database.
CREATETS	The CREATE TABLESPACE statement, to create table spaces in the database.
DISPLAYDB	The DISPLAY DATABASE command, to display the database status.
DROP	DROP and ALTER DATABASE, to drop or alter the database.

Table 3.2: Explicit privileges (continued)	
Privilege	**Provides this usage**
IMAGCOPY	The QUIESCE, COPY, and MERGECOPY utilities, to prepare for, make, and merge copies of table spaces in the database; and the MODIFY RECOVERY utility, to remove records of copies.
LOAD	The LOAD utility, to load tables in the database.
RECOVERDB	The RECOVER, REBUILD INDEX, and REPORT utilities, to recover objects in the database and report their recovery status.
REORG	The REORG utility, to reorganize objects in the database.
REPAIR	The REPAIR and DIAGNOSE utilities (except REPAIR DBD and DIAGNOSE WAIT) to generate diagnostic information about, and repair data in, objects in the database.
STARTDB	The START DATABASE command, to start the database.
STATS	The RUNSTATS, CHECK, LOAD, REBUILD INDEX, REORG INDEX, and REORG TABLESPACE utilities, to gather statistics and check indexes and referential constraints for objects in the database and delete unwanted statistics history records from the corresponding catalog tables.
STOPDB	The STOP DATABASE command, to stop the database.
Subsystem	
ARCHIVE	The ARCHIVE LOG command, to archive the current active log; the DISPLAY ARCHIVE command, to give information about input archive logs; the SET LOG command, to modify the checkpoint frequency specified during installation; and the SET ARCHIVE command, to control allocation and deallocation of tape units for archive processing.
BINDADD	The BIND subcommand with the ADD option, to create new plans and packages.
BINDAGENT	The BIND, REBIND, and FREE subcommands and the DROP PACKAGE statement, to bind, rebind, or free a plan or package, or to copy a package, on behalf of the grantor. The BINDAGENT privilege is intended for separation of function, not for added security. A bind agent with the EXECUTE privilege might be able to gain all the authority of the grantor of BINDAGENT.
BSDS	The RECOVER BSDS command, to recover the bootstrap data set.
CREATEALIAS	The CREATE ALIAS statement, to create an alias for a table or view name.
CREATEDBA	The CREATE DATABASE statement, to create a database and have DBADM authority over it.
CREATEDBC	The CREATE DATABASE statement, to create a database and have DBCTRL authority over it.
CREATESG	The CREATE STOGROUP statement, to create a storage group.
CREATETMTAB	The CREATE GLOBAL TEMPORARY TABLE statement, to define a created temporary table.
DEBUGSESSION	The DEBUGINFO connection attribute, to control debug session activity for native SQL and Java stored procedures.

Table 3.2: Explicit privileges (continued)	
Privilege	**Provides this usage**
DISPLAY	The DISPLAY ARCHIVE, DISPLAY BUFFERPOOL, DISPLAY DATABASE, DISPLAY LOCATION, DISPLAY LOG, DISPLAY THREAD, and DISPLAY TRACE commands, to display system information.
MONITOR1	Receive trace data that is not potentially sensitive.
MONITOR2	Receive all trace data.
RECOVER	The RECOVER INDOUBT command, to recover threads.
STOPALL	The STOP DB2 command, to stop DB2.
STOSPACE	The STOSPACE utility, to obtain data about space usage.
TRACE	The START TRACE, STOP TRACE, and MODIFY TRACE commands, to control tracing.
	Usage
USE OF BUFFERPOOL	A buffer pool.
USAGE ON JAR	A Java class.
USAGE ON SEQUENCE	A sequence.
USE OF STOGROUP	A storage group.
USE OF TABLESPACE	A table space.
	Schema
CREATEIN	Create distinct types, user-defined functions, triggers, and stored procedures in the designated schemas.
ALTERIN	Alter user-defined functions or stored procedures, or specify a comment for distinct types, user-defined functions, triggers, and stored procedures in the designated schemas.
DROPIN	Drop distinct types, user-defined functions, triggers, and stored procedures in the designated schemas.
	Distinct type
USAGE ON DISTINCT TYPE	A distinct type.
	Routine
EXECUTE ON FUNCTION	A user-defined function.
EXECUTE ON PROCEDURE	A stored procedure.
	Sequence Object
ALTER	A sequence object.

GRANTing and REVOKEing Privileges

The privileges in Table 3.2 must be GRANTed to an authorization ID. The GRANT and REVOKE statements are part of the SQL language known as *Data Control Language* (*DCL*). Let's take a look at some examples of granting and revoking privileges.

To grant the ID DB2EXPT the ability to select data from a particular table, you would execute the following SQL statement.

```
GRANT SELECT ON DSN8910.EMP TO DB2EXPT
```

To grant DB2EXPT the ability to BIND packages to the DB2SAMPL collection, you would execute this statement:

```
GRANT BIND ON PACKAGE DB2SAMPL.* TO DB2EXPT
```

To grant everyone the ability to select, update, insert, or delete data from a particular table, you'd execute this statement:

```
GRANT ALL ON DSN8910.EMP TO PUBLIC
```

Note that the keyword PUBLIC lets any user have the granted privilege.

To take away everyone's delete authority from a particular table, you would execute this statement:

```
REVOKE DELETE ON DSN8910.EMP FROM PUBLIC
```

Revoking a privilege from a user can also cause that privilege to be revoked from other users. This type of revoke is called a *cascade revoke*.

Related and Inherited Privileges

DB2 defines sets of related privileges that are identified by administrative authorities (which we examine the next). This grouping makes it easier to administer authority because instead of having to grant several individual privileges to an ID, you can simply grant the administrative authority—which includes all applicable privileges. Some privileges are also inherited with object ownership.

Authorities

An *administrative authority* is a set of privileges, often covering a related set of objects. Authorities often include privileges that aren't explicit, have no name, and cannot be specifically granted—for example, the ability to terminate any utility job, which is included in the SYSOPR authority. The nine DB2 administrative authorities are

- Installation SYSADM
- SYSCTRL
- SYSADM
- SYSOPR
- Installation SYSOPR
- PACKADM
- DBMAINT
- DBCTRL
- DBADM

Table 3.3 describes the capabilities and privileges of each authority.

The DBCTRL authority provides a good way to assign all the necessary privileges a DBA needs to perform his or her duties without granting the ability to access the data itself.

Table 3.3: DB2 administrative authorities

Authority	Capabilities	Privileges
Installation SYSADM	Assigned during DB2 installation, this authority has all the privileges of the SYSADM authority. In addition: • Authority is *not* recorded in the DB2 catalog. The catalog need not be available to check installation SYSADM authority. (The authority outside the catalog is crucial. If the catalog table space SYSDBAUT is stopped, for example, DB2 can't check the authority to start it again. Only an installation SYSADM can start it.) • No ID can revoke this authority; it can be removed only by changing the module that contains the subsystem initialization parameters (typically DSNZPARM). SYSADM IDs can also • run the CATMAINT utility • access DB2 when the subsystem is started with ACCESS(MAINT) • start databases DSNDB01 and DSNDB06 when they are stopped or in restricted status • run the DIAGNOSE utility with the WAIT statement • start and stop the database containing the application registration table (ART) and the object registration table (ORT)	All privileges of all the authorities.
SYSCTRL	This authority has almost complete control of the DB2 subsystem but *cannot* access user data directly unless granted the privilege to do so. Designed for administering a system containing sensitive data, SYSCTRL can • act with installation SYSOPR authority (when the catalog is available) or with DBCTRL authority over any database • run any allowable utility on any database • issue a COMMENT ON, LABEL ON, or LOCK TABLE statement for any table • create a view for itself or others on any catalog table • create tables and aliases for itself or others • bind a new plan or package, naming any ID as the owner	System privileges: • BINDADD • BINDAGENT • BSDS • CREATEALIAS • CREATEDBA • CREATEDBC • CREATESG • CREATETMTAB • MONITOR1 • MONITOR2 • STOSPACE Privileges on all tables: • ALTER • INDEX • REFERENCES • TRIGGER

Table 3.3: DB2 administrative authorities (continued)

Authority	Capabilities	Privileges
	Without additional privileges, SYSCTRL cannot • execute SQL Data Manipulation Language (DML) statements on user tables or views • run plans or packages • set the current SQL ID to a value that is not one of its primary or secondary IDs • start or stop the database containing the ART and ORT • act fully as SYSADM or as DBADM over any database • access DB2 when the subsystem is started with ACCESS(MAINT) • revoke a privilege granted by another ID *Note*: SYSCTRL authority is intended for separation of function, not for added security.	Privileges on catalog tables: • DELETE • INSERT • SELECT • UPDATE Privileges on all plans: • BIND Privileges on all packages: • BIND • COPY Privileges on all collections: • CREATE IN Privileges on all schemas: • ALTERIN • CREATE IN • DROPIN Use privileges on: • BUFFERPOOL • STOGROUP • TABLESPACE
SYSADM	This authority includes SYSCTRL, plus access to all data. SYSADM can • use all privileges of the DBADM authority over any database • use EXECUTE and BIND on any plan or package and use COPY on any package • use privileges over views owned by others • set the current SQL ID to any valid value, whether it is currently a primary or secondary authorization ID • create and drop synonyms and views for others on any table • use any valid value for OWNER in BIND or REBIND • drop database DSNDB07 • grant any of the privileges listed above to others Holders of SYSADM authority can also drop or alter any DB2 object except system databases, issue a COMMENT ON or LABEL ON statement for any table or view, and terminate any utility job; however, SYSADM cannot specifically grant these privileges.	All privileges held by SYSCTRL and DBADM Plan privileges: • EXECUTE Package privileges: • BIND • COPY Routine privileges: • EXECUTE Distinct type and sequence privileges: • USAGE Debug privileges: • DEBUGSESSION

Table 3.3: DB2 administrative authorities (continued)

Authority	Capabilities	Privileges
SYSOPR	This authority can • issue most DB2 commands except ARCHIVE LOG, START DATABASE, STOP DATABASE, and RECOVER BSDS • terminate any utility job • execute the DSN1SDMP utility	System privileges: • DISPLAY • RECOVER • STOPALL • TRACE Privileges on routines: • START DISPLAY • STOP
Installation SYSOPR	This authority is assigned during DB2 installation and has the following privileges in addition to those of SYSOPR: • Authority is not recorded in the DB2 catalog. The catalog need not be available to check installation SYSOPR authority. • No ID can revoke the authority; it can be removed only by changing the module that contains the subsystem initialization parameters (typically DSNZPARM). The SYSOPR authority can • access DB2 when the subsystem is started with ACCESS(MAINT) • run all allowable utilities on the directory and catalog databases (DSNDB01 and DSNDB06) • run the REPAIR utility with the DBD statement • start and stop the database containing the ART and ORT • issue dynamic SQL statements that aren't controlled by the DB2 governor • issue a START DATABASE command to recover objects that have logical page list (LPL) entries or group buffer pool recovery-pending status SYSOPR IDs cannot change the access mode.	All privileges held by SYSOPR System privileges: • ARCHIVE • STARTDB (cannot change access mode)
PACKADM	This authority has all package privileges on all packages in specific collections, or on all collections, plus the CREATE IN privilege on those collections. If the installation option BIND NEW PACKAGE is set to BIND, PACKADM also has the privilege to add new packages or new versions of existing packages.	Privileges on a collection: • CREATE IN Privileges on all packages in the collection: • BIND • COPY • EXECUTE

Table 3.3: DB2 administrative authorities (continued)

Authority	Capabilities	Privileges
DBMAINT	This authority is granted for a specific database, in which the ID can create certain objects, run certain utilities, and issue certain commands. It can use the TERM UTILITY command to terminate all utilities except DIAGNOSE, REPORT, and STOSPACE on the database.	Privileges on one database: • CREATETAB • CREATETS • DISPLAYDB • IMAGCOPY • STARTDB • STATS • STOPDB
DBCTRL	In addition to DBMAINT privileges, the DBCTRL authority can run utilities that can change the data.	All privileges held by DBMAINT on a database Privileges on one database: • DROP • LOAD • RECOVERDB • REORG • REPAIR
DBADM	In addition to the privileges held by DBCTRL over a specific database, DBADM has privileges to access any of its tables through SQL statements. It can also drop and alter any table space, table, or index in the database and issue a COMMENT ON, LABEL ON, or LOCK TABLE statement for any table. If the value of field DBADM CREATE VIEW on installation panel DSNTIPP was set to YES during DB2 installation, a user with DBADM authority can create a view for another user ID on any table or combination of tables and views in a database.	All privileges held by DBCTRL on a database Privileges on tables and views in one database: • ALTER • DELETE • INDEX • INSERT • REFERENCES • SELECT • TRIGGER • UPDATE

GRANTing and REVOKEing Authorities

The authorities in Table 3.3 must be GRANTed to an authorization ID. Just as with privileges, you can accomplish this task using the GRANT and REVOKE statements. Let's look at a couple of examples of granting and revoking authorities.

To grant DBADM authority on the DSN8D91A database to ID DB2EXPT, you'd issue this statement:

```
GRANT DBADM ON DSN8D91A TO DB2EXPT
```

To remove PACKADM authority from DB2EXPT, you'd issue this statement:

```
REVOKE PACKADM FROM DB2EXPT
```

WITH GRANT OPTION

If you GRANT an authority using the WITH GRANT option, the holder can GRANT the privileges contained in that authority to others. To grant DB2EXPT DBADM authority on the DSN8D91A database and permit DB2EXPT to give this authority to others, you'd issue this statement:

```
GRANT DBADM ON DSN8D91A TO DB2EXPT WITH GRANT OPTION
```

if the DBADM authority is ever revoked from DB2EXPT, any ID that has been granted DBADM from this ID will also be revoked automatically.

Ownership

Implicit privileges are included with ownership of an object. When you create DB2 objects (other than plans and packages) by issuing SQL CREATE statements in which you name the object, you establish ownership. The owner implicitly holds certain privileges over the owned object.

The privileges inherent in the ownership of an object cannot be revoked.

Unqualified Objects

If an object name is unqualified, the object ownership established depends on the type of object. Ownership of tables, views, indexes, aliases, and synonyms with unqualified names is established differently from ownership of user-defined

functions, stored procedures, distinct types, sequences, and triggers with unqualified names.

If the name of a table, view, index, alias, or synonym is unqualified, you establish the object's ownership in these ways:

- If the CREATE statement is issued dynamically (via SPUFI or the Query Management Facility), the owner of the created object is the current SQL ID of the issuer. That ID must have the privileges necessary to create the object.
- If the CREATE statement is issued statically (by executing a plan or package that contains it), the ownership of the created object depends on the option used for the bind operation. You can bind the plan or package with the QUALIFIER option, the OWNER option, or both.
  - » With the QUALIFIER option only, the QUALIFIER is the owner of the object. The QUALIFIER option lets the binder name a qualifier to use for all unqualified names of tables, views, indexes, aliases, or synonyms that appear in the plan or package.
  - » With the OWNER option only, the OWNER is the owner of the object.
  - » With both the QUALIFIER option and the OWNER option, the QUALIFIER is the owner of the object.
  - » If neither option is specified, the binder of the plan or package is implicitly the object owner.

The plan or package owner must have all required privileges on the objects designated by the qualified names.

You establish the ownership a user-defined function, stored procedure, distinct type, sequence, or trigger in the following ways:

- If the CREATE statement is issued dynamically, the owner of the created object is the current SQL ID of the issuer. That ID must have the privileges necessary to create the object.
- If the CREATE statement is issued statically (by running a plan or package that contains it), the owner of the object is the plan or package owner. You can use the OWNER bind option to explicitly name the object owner. If the OWNER bind option is not specified, the binder of the package or plan is implicitly the object owner.

The implicit qualifier is determined for an unqualified user-defined function, stored procedure, distinct type, sequence, or trigger by the name in the dynamic statements or the PATH bind option in static statements. The owner of a JAR that is used by a stored procedure or user-defined function is the current SQL ID of the process that performs the INSTALL_JAR function.

Qualified Objects

If an object name is qualified, the way ownership of the object is established depends, again, on the type of object.

For tables, views, indexes, aliases, or synonyms created with a qualified name, the qualifier is the owner of the object and is the schema name. The schema name identifies the schema to which the object belongs. All objects qualified by the same schema are related.

If you create a distinct type, user-defined function, stored procedure, sequence, or trigger with a qualified name, the qualifier will also be the schema name. The schema name identifies the schema to which the object belongs. You can think of all objects that are qualified by the same schema name as a group of related objects. Unlike with other objects, however, this qualifier doesn't identify the owner of the object. You establish ownership of a distinct type, user-defined function, stored procedure, or trigger as follows:

- If you issue the CREATE statement dynamically, the owner of the created object is your current SQL ID. That ID must have the privileges necessary to create the object.

- If you issue the CREATE statement statically (by running a plan or package that contains it), the owner of the object is the plan or package owner. You can use the OWNER bind option to explicitly name the object owner. If the OWNER bind option is not used, the binder of the package or plan is the implicit object owner.

For more information about schemas, see Chapter 15.

Objects Within a Trusted Context

Roles can help simplify administration by serving as owners of objects. If the owner of an object is an authorization ID and you need to transfer the ownership to another ID, you must first drop the object first and then re-create it with the new authorization ID as the owner. If the owner is a role, these steps are unnecessary because all the users that are associated with that role have the owner privilege.

The definition of a trusted context determines the ownership of objects that are created in the trusted context. Assume you issue the CREATE statement dynamically and define the trusted context using the ROLE AS OBJECT OWNER clause. In this case, the associated role is the owner of the objects, regardless of whether the objects are explicitly qualified.

In contrast, assume you issue the CREATE statement statically and the plan or package is bound in the trusted context with the ROLE AS OBJECT OWNER clause. In this case, the role that owns the plan or package also owns the created objects, regardless of whether the objects are explicitly qualified.

Privileges of Ownership by Object

Table 3.4 lists the privileges that are inherited with ownership of an object.

Table 3.4: Privileges inherited with object ownership

Object type	Implicit privileges of ownership
Storage group	• ALTER or DROP the storage group • Name the storage group in the USING clause of a CREATE INDEX or CREATE TABLESPACE statement
Database	• DBCTRL or DBADM authority over the database, depending on the privilege (CREATEDBA or CREATEDBC) used to create the database. DBCRTL authority does not include the privilege to access data in tables in the database
Table space	• ALTER or DROP the table space • Name the table space in the IN clause of a CREATE TABLE statement
Table	• ALTER or DROP the table or any indexes on it • Use LOCK TABLE, COMMENT ON, or LABEL on the table • CREATE an index or view on the table • SELECT or UPDATE any row or column • INSERT or DELETE any row • Use the LOAD utility for the table • Define referential constraints on any table or set of columns • CREATE a trigger on the table
Index	• ALTER or DROP the index
View	• DROP, COMMENT ON, or LABEL the view, or SELECT any row or column • UPDATE any row or column • INSERT or DELETE any row (if the view is not read-only)
Synonym	• USE or DROP the synonym
Trusted context	• CREATE, ALTER, COMMIT, REVOKE, or COMMENT ON the trusted context
Package	• BIND, REBIND, FREE, COPY, DROP, EXECUTE, or DROP the package
JAR	• REPLACE, USE, or DROP the JAR
Plan	• BIND, REBIND, FREE, or EXECUTE the plan
Alias	• DROP the alias
Distinct type	• USE or DROP a distinct type
Role	• CREATE, ALTER, COMMIT, DROP, or COMMENT ON the role
Sequence	• ALTER, COMMENT ON, USE, or DROP the sequence
User-defined functions	• EXECUTE, ALTER, DROP, START, STOP, or DISPLAY a user-defined function
Stored procedure	• EXECUTE, ALTER, DROP, START, STOP, or DISPLAY a stored procedure

Plan or Package Ownership

An application plan or a package can take many actions on many tables, all of them requiring one or more privileges. The owner of the plan or package must hold every required privilege. Another ID can execute the plan with just the EXECUTE privilege. In this way, another ID can exercise all the privileges used in validating the plan or package, but only within the restrictions imposed by the SQL statements in the original program.

The executing ID can use some of the owner's privileges, within limits. If the privileges are revoked from the owner, the plan or the package is invalidated; it must be rebound, and the new owner must have the required privileges.

The BIND and REBIND subcommands create or change an application plan or package. On either subcommand, use the OWNER option to name the owner of the resulting plan or package. When naming an owner, keep the following points in mind.

- If you use the OWNER option:
  - » Any user can name the primary or any secondary ID.
  - » An ID with the BINDAGENT privilege can name the grantor of that privilege.
  - » An ID with SYSADM or SYSCTRL authority can name any authorization ID on a BIND command, but not on a REBIND command.
- If you omit the OWNER option:
  - » On a BIND command, the primary ID becomes the owner.
  - » On a REBIND command, the previous owner retains ownership.

Unqualified Names

A plan or package can contain SQL statements that use unqualified table and view names. For static SQL, the default qualifier for these names is the owner of the plan or package. However, you can use the QUALIFIER option of the BIND command to specify a different qualifier.

For plans or packages that contain static SQL, using the BINDAGENT privilege and the OWNER and QUALIFIER options gives you considerable flexibility in performing bind operations. For plans or packages that contain dynamic SQL, the DYNAMICRULES behavior determines how DB2 qualifies unqualified object names.

For unqualified distinct types, user-defined functions, stored procedures, sequences, and trigger names in dynamic SQL statements, DB2 finds the schema name to use as the qualifier by searching schema names in the CURRENT PATH special register. For static statements, the PATH bind option determines the path that DB2 searches to resolve unqualified distinct types, user-defined functions, stored procedures, and trigger names.

However, an exception exists for ALTER, CREATE, DROP, COMMENT ON, GRANT, and REVOKE statements. For static SQL, specify the qualifier for these statements in the QUALIFIER bind option. For dynamic SQL, the qualifier for these statements is the authorization ID of the CURRENT SQLID special register.

Trusted Context

You can issue the BIND and REBIND commands in a trusted context with the ROLE AS OBJECT OWNER clause to specify the ownership of a plan or package. In this trusted context, you can specify only a role, not an authorization ID, as the OWNER of a plan or package. If you specify the OWNER option, the specified role becomes the owner of the plan or package. If you don't specify the OWNER option, the role that is associated with the binder becomes the owner. If you omit the ROLE AS OBJECT OWNER clause for the trusted context, the current rules for plan and package ownership apply.

If you want a role to own the package at a remote DB2 data server, you need to define the role ownership in the trusted context at the remote server. Be sure to establish the connection to the remote DB2 as trusted when binding or rebinding the package at the remote server.

If you specify the OWNER option in a trusted connection during the remote BIND processing, the outbound authorization ID translation is not performed for the OWNER.

If the plan owner is a role and the application uses a package bound at a remote DB2 for z/OS data server, the privilege of the plan owner to execute the package is not considered at the remote DB2 server. The privilege set of the authorization ID (either the package owner or the process runner, depending on the DYNAMICRULES behavior) at the DB2 for z/OS data server must have the EXECUTE privilege on the package at the DB2 data server.

Plan Execution Authorization

The plan or package owner must have authorization to execute all static SQL statements that are embedded in the plan or package. These authorizations do not need to be in place when the plan or package is bound, nor do the objects that are referred to need to exist at that time.

A bind operation always checks whether a local object exists and whether the owner has the required privileges on it. Any failure results in a message. To choose whether the failure prevents the bind operation from being completed, use the VALIDATE option of the BIND PLAN and BIND PACKAGE subcommands and also the SQLERROR option of BIND PACKAGE. If you permit the operation to be completed, the checks occur again at run time. The corresponding checks for remote objects are always made at run time.

Authorization to execute dynamic SQL statements is also checked at run time.

To include a package in a plan's PKLIST, the owner will need to be given execute authority on the package.

For more information about plans and packages, see Chapter 11.

Catalog Table Information for Object Access

Table 3.5 provides information about the authorities and privileges currently held on various objects in the DB2 subsystem.

Table 3.5: DB2 catalog table authorities and privileges	
DB2 catalog table	**Authorities/Privileges**
SYSIBM.SYSCOLAUTH	Update column authority
SYSIBM.SYSDBAUTH	Database privileges
SYSIBM. SYSPACKAUTH	Package privileges
SYSIBM. SYSPLANAUTH	Plan privileges
SYSIBM.SYSRESAUTH	Buffer pool, storage group, collection, table space, and distinct type use privileges
SYSIBM. SYSROUTINEAUTH	User-defined functions and stored procedure privileges
SYSIBM. SYSSCHEMAAUTH	Schema privileges
SYSIBM. SYSEQUENCEAUTH	Sequence object privileges
SYSIBM.SYSTABAUTH	Tables and view privileges
SYSIBM. SYSUSERAUTH	System authorities

Controlling Access with Views

By using views, you can control what data a user can see, whether it be certain columns, certain rows, or even a combination of rows and columns. Views thus give you a way, in addition to granting privileges and authorities, to further restrict access to data. You implement this type of access control by creating a view that lets users see only certain columns or rows and then permitting them access to only the view, not the base table.

The following example permits the user of the view to see the names of employees who work in department D01.

```
CREATE VIEW EMPVIEW
AS
SELECT FIRSTNME, LASTNAME
FROM DSN8910.EMP
WHERE WORKDEPT = 'D01'
```

Multilevel Security

Multilevel security enables a more granular approach to setting security, combining hierarchical and categorical security schemes. Organizations can use this type of security to prevent individuals from accessing data at a higher security level or from declassifying data.

DB2 supports multilevel security at the row level. With row-level security, the system restricts individual user access to a specific set of rows in a table. This security method requires Fz/OS 1.5 RACF at a minimum.

The security enforcement occurs automatically at statement run time and lets you perform new security checks that are difficult to express using SQL views or queries. Multilevel security doesn't rely on special views or database variables, and the controls are consistent and integrated across the system.

User security classification is maintained in the RACF security database only.

To support multilevel security, the DB2 tables require a new column, defined as AS SECURITY LABEL. This column contains the security label. Every row has a specific security label; these values correspond to security label definitions. For each accessed row, DB2 calls the RACF Security Exit to check authorization. If access is authorized, normal data access is permitted; otherwise, data is not returned. To reduce overhead, the security labels are cached.

Security Function DB2_SECURE_VAR

DB2 provides a way to feed external security information into SQL. The variables are set by the connection/sign-on exit routines. Built-in function DB2_SECURE_VAR lets you retrieve the value for a variable. You can use this variable in views, triggers, stored procedures, functions, and constraints to enforce security policies.

Here's an example of using function DB2_SECURE_VAR in a view.

```
CREATE VIEW MY_DATA AS
         SELECT *
           FROM SHARED_DATA
          WHERE COL_OWNER
              = DB2_SECURE_VAR('SEC_OWNER')
```

Auditing

This chapter answers some fundamental auditing questions, the following two before foremost among them:

- Who is privileged to access what objects?
- Who has actually accessed the data?

The DB2 catalog holds the answer to the first question: it contains a primary audit trail for the DB2 subsystem. Most of the catalog tables describe the DB2 objects, such as tables, views, table spaces, packages, and plans. Several other tables (those with "AUTH" in their name) hold records of every grant of a privilege or authority on different types of objects. Each grant record contains the name of the object, the ID that received the privilege, the ID that granted it, the time of the grant, and other information. You can retrieve data from the catalog tables by writing SQL queries.

Audit Trace

Another primary audit trail for DB2 is the audit trace. The trace can record changes in authorization IDs for a security audit as well as changes made to the structure of data (e.g., dropping a table) or to data values (e.g., updating or inserting records) for an audit of data access. You can also use the audit trace to track access attempts by unauthorized IDs, the results of GRANT and REVOKE statements, the mapping of Kerberos security tickets to RACF IDs, and other activities of interest to auditors.

The audit trace can answer the question of who has accessed data. When started, the trace creates records of actions of certain types and sends them to a named destination. From these records, you can obtain information such as

- the ID that initiated an activity
- the LOCATION of the ID that initiated the activity (if the access was initiated from a remote location)
- the type of activity and the time it occurred
- the DB2 objects affected

- whether access was denied
- who owns a particular plan and package

Using the audit trace, you can also determine which primary ID is responsible for the action of a secondary ID when that information might not appear in the catalog.

Whether a request comes from a remote location or from the local DB2, it can be audited. For a remote request, the authorization ID on a trace record is the ID that is the final result of any outbound translation, inbound translation, or activity of an authorization exit routine — that is, it is the same ID to which you've granted access privileges for your data. Requests from your location to a remote DB2 are audited only if an audit trace is active at the remote location. The trace output appears only in the records at that location.

Trace Details

The audit trace doesn't record everything. The actual changed data is recorded in the DB2 log. If an agent or transaction accesses a table more than once in a single unit of recovery, the trace records only the first access, and then only if you've started the audit trace for the appropriate class of events.

Some utilities are not audited. The first access of a table by the LOAD utility is audited, but access by COPY, RECOVER, and REPAIR is not. Access by standalone utilities, such as DSN1CHKR and DSN1PRNT, is not audited. (For more information about these DB2 utilities, see Chapter 7.)

Everything comes at a cost. Auditing does impose some overhead and can produce more data than necessary.

When you start the trace, you choose the events to audit by supplying one or more numbers to identify classes of events. Trace records are limited to 5,000 bytes, so descriptions that contain long SQL statements may be truncated. Table 3.6 lists the

available classes and the events they include. (For more information about trace classes, see Chapter 17.)

The audit trace does not audit DB2 commands.

Table 3.6: Audit trace event classes

Class	Events traced
1	Access attempts that DB2 denies because of inadequate authorization. This class is the default.
2	Explicit GRANT and REVOKE statements and their results. This class does not include implicit grants and revokes.
3	CREATE, ALTER, and DROP operations affecting audited tables, and their results. This class includes the dropping of a table caused by DROP TABLESPACE or DROP DATABASE and the creation of a table with AUDIT CHANGES or AUDIT ALL. The trace audits ALTER TABLE statements only when they change the AUDIT option for the table.
4	Changes to audited tables. Only the first attempt to change a table, within a unit of recovery, is recorded. (If the agent or the transaction issues more than one COMMIT statement, the number of audit records increases accordingly.) The changed data isn't recorded, only the attempt to make a change. If the change is not successful and is rolled back, the audit record remains; it is not deleted. This class includes access by the LOAD utility. The trace also audits accesses to a dependent table that are caused by attempted deletions from a parent table. The audit record is written even if the delete rule is RESTRICT, which prevents the deletion from the parent table. The audit record is also written when the rule is CASCADE or SET NULL, which can result in deletions cascading to the dependent table.
5	All read accesses to tables identified as AUDIT ALL. As in class 4, only the first access within a DB2 unit of recovery is recorded, and references to a parent table are audited.
6	The bind of static and dynamic SQL statements of the following types: • INSERT, UPDATE, DELETE, CREATE VIEW, and LOCK TABLE statements for audited tables. Except for the values of host variables, the audit record contains the entire SQL statement. • SELECT statements to tables identified as AUDIT ALL. Except for the values of host variables, the audit record contains the entire SQL statement.
7	Assignment or change of an authorization ID through an exit routine (default or user-written) or SET CURRENT SQLID statement, through an outbound or inbound authorization ID translation, or because the ID is being mapped to an RACF ID from a Kerberos security ticket.
8	The start of a utility job, and the end of each phase of the utility.
9	The writing of various types of records to IFCID 0146 by the IFI WRITE function.
10	CREATE and ALTER TRUSTED CONTEXT statements, establish trusted connection information, and switch user information.

Auditing Specific IDs or Roles

You can start the audit trace for a particular plan name, a particular primary authorization ID, or a combination of both. Having audit traces on at all times can be useful for IDs with SYSADM authority, for example, because they have complete access to every table. If you have a network of DB2 subsystems, you might need to trace multiple authorization IDs for those users whose primary authorization ID is translated several times.

By using the ROLE and XROLE filters, you can also start traces for a particular role in a trusted context.

Starting/Stopping the Trace

To start the audit trace, you execute the –START TRACE command. The following example starts a trace that audits data changes and captures the text of any dynamic SQL.

```
-START TRACE (AUDIT) DEST (SMF)
COMMENT ('Trace data changes; include text of dynamic DML statements.')
```

To stop this trace, issue the –STOP TRACE command:

```
-STOP TRACE (AUDIT)
```

This command simply stops the last trace started. If more than one trace is executing, you can use the –DISPLAY TRACE command to identify a particular trace by number. (For more information about DB2 commands, see Chapter 2.)

You can configure the audit trace to start automatically when DB2 is started by using an option on the DSNTIPN panel when you install DB2. You can set the AUDIT TRACE option to NO, YES, or a list of audit trace classes.

Auditing a Table

For the audit trace to be effective at the table level, you must first choose, by specifying an option of the CREATE or ALTER statement, whether to audit the table. This example shows how to indicate that you want to audit changes.

```
CREATE TABLE DSN8910.EMP
(EMPNO CHAR(6) NOT NULL
  ...
IN DSN8D91A.DSN8S91E
AUDIT CHANGES
```

Possible AUDIT values are CHANGES, ALL, and NONE, with the default being NONE (no auditing). To turn off auditing at the table level, you'd simply perform an ALTER specifying AUDIT NONE.

You cannot audit auxiliary tables or catalog tables.

Summary

In this chapter, we discussed several topics related to data access. We covered security with respect to the subsystem, data sets, and DB2. DB2 lets you control subsystem security in a variety of ways, such as via CICS, IMS, and Kerberos and RACF. In some situations, you also need to consider securing access at the data-set level because DB2 stores its data into individual data sets that can be accessed outside DB2.

We discussed authorizations IDs (both primary and secondary), roles, and how both are assigned. DB2 provides several administrative authority levels: SYSADM, SYSCTRL, DBADM, DBCTRL, PACKADM, and so on. We discussed each of these authority types and the privileges they possess. Object ownership also comes with inherited authorities and privileges that can be granted to other authorizations IDs.

We examined the granting and revoking of database object privileges using the GRANT and REVOKE SQL statements.

We talked in detail about the DB2 audit trace, which lets you carefully monitor critical tables to see who is manipulating data or, in some very sensitive cases, who is simply trying to access data.

The trusted context and roles are new to DB2 9 and provide another level of security and manageability for your databases and applications. All these levels of security can work together to keep your data and your subsystem safe.

Additional Resources

IBM DB2 9 Administration Guide (SC18-9840)
IBM DB2 9 SQL Reference (SC18-9854)

Practice Questions

Question 1

DBA1 needs to be able to create tables in a database and be able to run periodic REORGs. However, DBA1 is not permitted to access the data or manipulate it. Which of the following authorities would be the most appropriate?

❍ A. SYSADM

❍ B. DBADM

❍ C. DBCTRL

❍ D. DBMAINT

Question 2

To limit exposure for an application, a company creates an APP1 trusted context and an APP1_DBA role. DBA1 was assigned to this role and all the objects. What happens if DBA1 leaves the company and the ID is removed?

❍ A. The objects and privileges of the role are untouched.

❍ B. All privileges need to be re-granted.

❍ C. All dependent privileges are cascade revoked.

❍ D. The objects need to be dropped and re-created.

Question 3

A user ID is required to attach a client to the current application/process connection to enable client application testing of native SQL or Java procedures that are executed within the session. Which of the following system privileges is required?

❍ A. TRACE

❍ B. MONITOR1

❍ C. MONITOR2

❍ D. DEBUGSESSION

Question 4

A DBA wants to create a new plan using the BIND PLAN PKLIST option and specifying individual packages. What authority/privilege must the DBA have for the operation to be successful?

❍ A. COPY to copy the individual packages

❍ B. EXECUTE authority on each package specified in the PKLIST

❍ C. CREATE IN to name the collection containing the individual packages

❍ D. BINDAGENT to bind all the individual packages on behalf of their owner

Question 5

Which level of authority is required to revoke a privilege that another ID has granted?

❍ A. DBADM

❍ B. DBCTRL

❍ C. SYSOPR

❍ D. SYSCTRL

Answers

Question 1

The correct answer is **C**, DBCTRL. This authority will let DBA1 create tables in a specified database as well as run some utilities, such as REORG, on the table spaces in the database. This authority does not permit SQL Data Manipulation Language (DML) to be executed against the objects in the database.

Question 2

The correct answer is **A**, the objects and privileges of the role are untouched. One benefit of roles is the fact that if the ID assigned to the role is removed, nothing happens to the objects or privileges assigned to the role.

Question 3

The correct answer is **D**, DEBUGSESSION. This is a new privilege with DB2 9 that provides the ability to control debug session activity for native SQL and Java stored procedures.

Question 4

The correct answer is **B**, EXECUTE authority on each package specified in the PKLIST. To be able to create a new plan with packages, you must have EXECUTE authority on the packages in the PKLIST.

Question 5

The correct answer is **D**, SYSCTRL. This is the only level of authority that can revoke a privilege granted by another ID.