

CHAPTER 15 Security

Oracle SQL By Example, Fourth Edition by Alice Rischert. Published by Prentice Hall. Copyright © 2008 by Pearson Education, Inc.

CHAPTER OBJECTIVES

In this chapter, you will learn about:

- ▶ Users
- ▶ Privileges
- ▶ Roles
- ▶ Synonyms

Oracle protects the data in the database by implementing security via users, roles, and privileges. The SQL language commands used to accomplish these security tasks are known as Data Control Language (DCL) commands.

Oracle provides several different ways to enforce access control to ensure that only authorized users can log in and access the appropriate data. You want to avoid situations in which a user can accidentally drop an important table or sidestep security rules. Every database user has certain

system privileges that determine the type of actions the user can perform, such as create tables, drop views, or create other users.

Object privileges avoid any wrongful data modifications to individual tables or columns. The owner of database objects can assign these object privileges that control exactly who can access what objects and to what extent.

System and object privileges can be grouped together into a *role*. Part of a database administrator's (DBA's) job is to set up the correct security for database users. It is vital that the database be properly protected against any wrongful actions and unauthorized access.

661
662

LAB 15.1 Users, Privileges, Roles, and Synonyms

LAB OBJECTIVES

After this lab, you will be able to:

- ▶ Create Users and Grant and Revoke Privileges
- ▶ Create and Use Synonyms
- ▶ Create User-Defined Roles

Many of the commands and actions discussed throughout this lab are carried out by a DBA or an authorized user. In this lab, you will gain a better understanding of the rights required to perform the various SQL commands against the database.

What Is a Schema?

A *schema* is a collection of objects (for example, tables, views, indexes, sequences, triggers, synonyms). Each schema is owned by a single user account with the same name; in fact, the two terms *schema* and *user account* are often used interchangeably.

You can list the types of objects in the STUDENT schema by querying the USER_OBJECTS data dictionary view.

```
SELECT DISTINCT object_type
  FROM user_objects
OBJECT_TYPE
-----
INDEX
SEQUENCE
...
VIEW

4 rows selected.
```

To see all the different types of objects available for the user accessing the database, query the ALL_OBJECTS view. The result set on your database may vary from this result, as various users may have different object types and different privileges.

662

663

```
SELECT DISTINCT object_type
FROM all_objects
OBJECT_TYPE
-----
...
INDEX
...
PACKAGE
PACKAGE BODY
PROCEDURE
SEQUENCE
SYNONYM
TABLE
TABLE PARTITION
TRIGGER
...

25 rows selected.
```

Special Users: SYSTEM and SYS

When an Oracle database is created, it comes with a number of default accounts. Two extremely important accounts are SYS and SYSTEM.

The SYS account is the most privileged user. It owns the data dictionary. If you drop any of the objects of the SYS schema, you endanger the critical operation of the Oracle database.

The SYSTEM account is automatically granted the DBA role for database administration tasks. This role is typically used to create regular user accounts or accounts with the DBA role. Oracle suggests that you create an administrative type of account after you create the database. You can then use this DBA account to perform daily administrative tasks, without having to use the SYS and SYSTEM accounts.

Creating Users

To log in to the Oracle database, a user must have a username, a password, and certain system privileges. A username is created with the CREATE USER command, which uses the following syntax.

```
CREATE USER user IDENTIFIED
    {BY password [REPLACE
oldPassword]
    | EXTERNALLY | GLOBALLY AS
'external name' }
    [{DEFAULT TABLESPACE tablespace |
```

```
TEMPORARY TABLESPACE {tablespace |
tablespace_group} |
    QUOTA {integer[K|M] | UNLIMITED}
ON tablespace
[[QUOTA {integer[K|M] | UNLIMITED}
ON tablespace]...] |
PROFILE profile |
PASSWORD EXPIRE |
ACCOUNT {LOCK|UNLOCK}
}]
```

663

664

To create a new user, first log in as a user with the rights to create other users (for example, an account with DBA privileges or the user SYSTEM).



Starting with Oracle 11g, passwords are, by default, case-sensitive.

The following statement creates a new user called MUSIC with the password listen.

```
CREATE USER music IDENTIFIED BY
listen
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp
    QUOTA 15 M ON users
```

User created.

The message User created indicates the successful creation of the user. The keywords DEFAULT TABLESPACE indicate where any of the user's objects are stored. Here the tablespace is called USERS. The TEMPORARY TABLESPACE keywords allow you to determine where any sorting of data that cannot be performed in memory is temporarily stored.

In preparation for creating users in your database, you must find out what tablespaces exist in your Oracle database. Query the USER_TABLESPACES or DBA_TABLESPACES data dictionary views with the following query.

```
SELECT tablespace_name
      FROM dba_tablespaces
      ORDER BY tablespace_name
```

You can also refer to the readme.pdf file on the companion Web site, located at www.oraclesqlbyexample.com, for an example of the creation of the STUDENT user.

Assigning a default tablespace to the user does not mean that the user can actually store objects in this tablespace. The QUOTA 15 M ON users clause allows the MUSIC user to use up to 15 MB on the USERS tablespace.

If the user will create objects, it is important that you specify an appropriate default tablespace. It is never good practice to use the SYSTEM tablespace as the default tablespace because the SYSTEM tablespace should only contain the data dictionary and other internal Oracle system-related objects. If you run out of space on this SYSTEM tablespace because a user uses up a lot of space, the system comes to a complete halt.

Changing the Password and Altering the User Settings

When an individual user's account settings need to change, such as the password or the default tablespace, the user can be altered. The syntax of the ALTER USER command is as follows.

```
ALTER USER {user {IDENTIFIED
    {BY password [REPLACE
oldPassword] |
    EXTERNALLY|GLOBALLY AS
'external name' }
    DEFAULT TABLESPACE tablespace |
    TEMPORARY TABLESPACE
{tablespace|tablespace_group} |
    QUOTA {integer [K|M] |
UNLIMITED} ON tablespace
```

664

665


```
    [[QUOTA {integer [K|M] |
UNLIMITED} ON tablespace]...] |
    PROFILE profile |
    DEFAULT ROLE
        {role [,role]...|ALL[EXCEPT
role [,role]...]|NONE} |
    PASSWORD EXPIRE |
    ACCOUNT {LOCK|UNLOCK}
}}
```

The following statement changes MUSIC's password from listen to tone and changes the default tablespace from USERS to USER_DATA.

```
ALTER USER music IDENTIFIED BY tone
    DEFAULT TABLESPACE USER_DATA
User altered.
```

If you are using SQL*Plus, type the SQL*Plus PASSWORD command at the SQL> prompt and then enter the old and new passwords.

OPERATING SYSTEM AUTHENTICATION

Instead of using Oracle's logon name, a user can be authenticated through the operating system account; this is done via the IDENTIFIED BY EXTERNALLY

password option. Operating system authentication is offered on some platforms (for example, UNIX, Windows), and a username and password do not need to be entered when connecting to SQL*Plus. Furthermore, the operating system controls password modifications and expirations. You can find out more about this topic in the *Oracle Platform Guide* for your operating system.

LOCKED ACCOUNTS

As part of the database installation process, Oracle creates a number of default user accounts that may be locked. You can use the ALTER USER command with the ACCOUNT UNLOCK option to unlock those accounts if your user community needs to use them.

Oracle has a large number of syntax options as part of the ALTER USER command. You will explore these different options throughout this lab.

Dropping Users

You use the following command to drop a user.

```
DROP USER music
User dropped.
```

665

The DROP USER command drops the user if the user does not own any objects. The syntax for the DROP USER command is as follows.

```
DROP USER user [CASCADE]
```

If you want to also drop the objects owned by the user, execute the DROP USER command with the CASCADE keyword.

```
DROP USER music CASCADE
```



If the objects and their data need to be preserved, be sure to first back up the data by using the Oracle Data Pump utility or any other reliable method.

What Are Privileges?

A *privilege* is a right to execute a particular type of SQL statement. There are two types of privileges: system privileges and object privileges. An example of a system privilege is the right to create a table or an index. A particular object privilege allows you to access an individual object, such as the privilege to select from the

INSTRUCTOR table, to delete from the ZIPCODE table, or to SELECT a number from a specific sequence.

SYSTEM PRIVILEGES

To establish a connection to the database, a user must be granted certain system privileges. These privileges are granted either individually or in the form of roles. A *role* is a collection of privileges.

Although the user MUSIC has been created, the user cannot start a session, as you see from the following error message. The user lacks the CREATE SESSION system privilege to log in to the database.

```
CONN music/tone
ERROR: ORA-01045: user MUSIC lacks
CREATE SESSION
privilege; logon denied
```

[Table 15.1](#) lists a few examples of individual system privileges that can be granted to a user. For example, if you have the CREATE TABLE privilege, you can create tables in your schema; if you have the CREATE ANY TABLE privilege, you can create tables in another user's schema. The CREATE TABLE privilege includes the CREATE INDEX privilege, but before you are allowed to create these objects, you must make sure you

have either been granted a quota on the individual tablespace on which you would like to place the object or the RESOURCE role, which provides you with unlimited space on all tablespaces.

The SELECT ANY TABLE privilege gives access to all tables, but it does not permit data dictionary access; instead, an object privilege called SELECT ANY DICTIONARY provides SELECT rights on the data dictionary views.

666

TABLE 15.1 Examples of System Privileges

	SYSTEM PRIVILEGE NAME
Session	CREATE SESSION
	ALTER SESSION
Table	CREATE TABLE
	CREATE ANY TABLE
	ALTER ANY TABLE
	DROP ANY TABLE
	SELECT ANY TABLE
	UPDATE ANY TABLE
	DELETE ANY TABLE
	FLASHBACK ANY TABLE
Index	CREATE ANY INDEX
	ALTER ANY INDEX
	DROP ANY INDEX
Sequence	CREATE SEQUENCE
	CREATE ANY SEQUENCE
	ALTER ANY SEQUENCE
	DROP ANY SEQUENCE
View	CREATE VIEW
	CREATE ANY VIEW

DROP ANY VIEW

OBJECT PRIVILEGES

Object privileges are granted for a particular object (for example, table, view, sequence). [Table 15.2](#) lists examples of object privileges.

TABLE 15.2 Examples of Object Privileges

OBJECT TYPE	PRIVILEGE	PURPOSE
TABLE	SELECT	The right to query from an individual table.
	INSERT	The right to add new rows into an individual table
	UPDATE	The right to change rows in an individual table. You can optionally specify to allow UPDATE rights only on individual columns.
	DELETE	The right to remove rows from an individual table.
	REFERENCES	The right to reference a table in a foreign key constraint.
	ALTER	The right to change table and column definitions.
	INDEX	The right to create indexes on the individual table.
	FLASHBACK	The right to flashback a table.
	ALL	All possible object privileges on a table.

667

668

SEQUENCE SELECT		The right to increment values from a sequence and retrieve current values.
ALTER		The right to change the sequence definition.
PL/SQL Stored Objects	EXECUTE	The right to execute any stored procedure, function, or package.

The GRANT Command

You give a system privilege or an object privilege to a user by using the GRANT command. You can grant privileges individually or through a role.

The syntax for granting system privileges is as follows.

```
GRANT {system_privilege|role|ALL
PRIVILEGES}
    [{system_privilege|role|ALL
PRIVILEGES}]...
TO {user|role|PUBLIC} [{user|role|
PUBLIC}]...
[WITH ADMIN OPTION]
```

The following statement grants the CREATE SESSION system privilege to the MUSIC user. This allows the MUSIC user to establish a session to the database.

```
GRANT CREATE SESSION TO music
```

Object privileges grant certain privileges on specific objects, such as tables, views, or sequences. You can grant object privileges to other users when you want them to have access to objects you created. You can also grant the user access to objects you do not own if the object's owner gives you permission to extend rights to others.

The following is the general syntax for granting object privileges.

```
GRANT {object_privilege|ALL
[PRIVILEGES]}
    [(column[,column]... )]
    [{object_privilege|ALL
[PRIVILEGES]}
    [(column[,column]... )]]...
ON objectname
TO {user|role|PUBLIC} [{user|role|
PUBLIC}]...
[WITH GRANT OPTION]
```

668

669

For example, the following statement connects as the STUDENT user account within SQL*Plus and grants the SELECT privilege on the COURSE table to the new user MUSIC.

```
CONN student/learn
```

Connected.

```
GRANT SELECT ON course TO music
```

Grant succeeded.

In this case, the STUDENT user is the grantor, and MUSIC is the grantee, the recipient of the privileges. Now the MUSIC user can query the COURSE table.

In addition to SELECT, other object privileges can be granted on a table, such as INSERT, UPDATE, DELETE, ALTER, INDEX, and REFERENCES (refer to [Table 15.2](#)). The ALTER privilege allows another user to change table definitions with the ALTER table command, the INDEX privilege allows the creation of indexes on the table, and the REFERENCES privilege allows the table to be referenced with a foreign key constraint. You can also grant all object privileges at once by using the GRANT ALL command.

Object privileges can be assigned to other database objects, such as sequences, packages, procedures, and functions. SELECT and ALTER privileges can be granted on sequences. Packages, procedures, and functions require the EXECUTE privilege if other users want to run these stored programs.

If an object, such as a table, is dropped and then re-created, the grants need to be reissued. This is not the

case if the object is replaced with the CREATE OR REPLACE keywords available for views and stored programs.

You can grant UPDATE and REFERENCES privileges on individual columns on a table. For example, to grant update on the columns COST and DESCRIPTION of the COURSE table, execute the following command.

```
GRANT UPDATE (cost, description) ON  
course TO music  
Grant succeeded.
```

Roles

A role is several privileges collected under one role name. Using roles aids in administration of multiple privileges to users. Oracle includes predefined roles; three popular ones that contain a number of different system privileges are CONNECT, RESOURCE, and DBA.

The CONNECT role contains the CREATE SESSION system privilege that allows a user to start a session. (In prior Oracle versions, this role could also create views, tables, and sequences, among other operations.)

The RESOURCE role allows a user to create tables and indexes on any tablespace and to create PL/SQL stored objects (for example, packages, procedures, functions).

The DBA role includes all system privileges. This role is usually granted to a user who performs database administration tasks. [Table 15.3](#) lists the system privileges associated with each role.

TABLE 15.3 The CONNECT, RESOURCE, and DBA Roles

ROLE	PURPOSE
CONNECT	Contains CREATE SESSION system privilege.
RESOURCE	Includes these system privileges: CREATE TABLE, CREATE SEQUENCE, CREATE TRIGGER, CREATE PROCEDURE, CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, and CREATE TYPE.
DBA	Includes all system privileges and allows them to be granted WITH ADMIN OPTION. This does not include the privilege to start up and shut down the database.

You can also query the ROLE_SYS_PRIVS and DBA_SYS_PRIVS data dictionary views to list the individual system privileges for each role, which may

change in future versions. For example, starting with Oracle 11g, the CREATE VIEW system privilege is no longer part of the standard RESOURCE role. If you want to grant this system privilege to a specific user, you need to grant it explicitly.

```
GRANT CREATE VIEW TO music
Grant succeeded.
```

When a user is granted a role, the user acquires all the privileges defined within the role. The following statement uses the two predefined Oracle roles CONNECT and RESOURCE to grant a number of system privileges to the new user.

```
GRANT CONNECT, RESOURCE TO music
Grant succeeded.
```

Extending Privileges to Others

To extend an object privilege to another user, you must be the owner of the object or have received the privilege to pass it on to others through WITH GRANT OPTION. You may also pass on the privilege if you have been granted the GRANT ANY OBJECT system privilege.

The following SQL statement grants all object privileges on the COURSE table to the MUSIC user. It also passes to the MUSIC user the ability to grant these privileges to

yet other users, using the WITH GRANT OPTION.
Here, MUSIC is the grantee but can become a grantor if the privilege is passed on to another user.

```
GRANT ALL ON course TO music WITH  
GRANT OPTION
```

Grant succeeded.

To allow users to pass on system privileges to other users, you must have been granted the system privilege with WITH ADMIN OPTION or have been granted the GRANT ANY PRIVILEGE system privilege. For example, after execution of the following statement, the user MUSIC can grant the CREATE SESSION system privilege to other users.

```
GRANT CREATE SESSION TO music WITH  
ADMIN OPTION
```

Grant succeeded.

If you want other users to pass on a role to others, you must have either created the role, have been granted the role through the WITH ADMIN OPTION, or been granted the GRANT ANY ROLE system privilege.

You can see which system privileges you received through a role by querying the Oracle data dictionary view ROLE_SYS_PRIVS. For granted system privileges, query the data dictionary views

670

671

USER_SYS_PRIVS or DBA_SYS_PRIVS. [Table 15.4](#) lists a number of data dictionary views you may find useful when trying to determine individual object privileges, system privileges, and roles.

TABLE 15.4 Useful Data Dictionary Views

DATA DICTIONARY VIEW	PURPOSE
SESSION_PRIVS	All current system privileges available to an individual user
USER_SYS_PRIVS	System privileges granted to the user
ROLE_SYS_PRIVS	System privileges received through a role
ROLE_TAB_PRIVS	Object privileges received through a role
USER_TAB_PRIVS	Object grants
USER_COL_PRIVS	Individual column grants
USER_TAB_PRIVS_RECD	Object privileges received by the user
USER_TAB_PRIVS_MADE	Object privileges made by the user

The REVOKE Command

Privileges can be taken away with the REVOKE command. Use the following syntax to revoke system privileges.

```
REVOKE {system_privilege|role|ALL
PRIVILEGES}
[, {system_privilege|role|ALL
PRIVILEGES}] ...
FROM {user|role|PUBLIC} [, {user|
role|PUBLIC}] ...
```

The next example shows how the RESOURCE role is revoked from the user MUSIC.

```
REVOKE RESOURCE FROM music
Revoke succeeded.
```

Object privileges can also be revoked as in the following statement.

```
REVOKE UPDATE ON course FROM music
Revoke succeeded.
```

The syntax for revoking object privileges is listed here.

```
REVOKE {object_privilege|ALL
[PRIVILEGES]}
```

671

672

```
[, {object_privilege|ALL  
[PRIVILEGES]}  
ON objectname  
FROM {user|role|PUBLIC} [, {user|  
role|PUBLIC}] ...  
[CASCADE CONSTRAINTS]
```

The **CASCADE CONSTRAINTS** clause is needed only if you revoke the **REFERENCES** or **ALL** privileges. The **REFERENCES** privilege allows you to create a referential integrity constraint based on another user's object. The **CASCADE CONSTRAINT** options drops any defined referential constraints when you revoke the **REFERENCES** privilege.

Object privileges granted using the **WITH GRANT OPTION** are revoked if the grantor's object privilege is revoked. For example, assume that **USER1** is granted **SELECT** privilege on the **COURSE** table, using the **WITH GRANT OPTION**, and grants the same privilege to **USER2**. If the **SELECT** privilege is revoked from **USER1**, then the revoke cascades to **USER2**.



Revoking object privileges cascades the REVOKE to other users. However, revoking system privileges does not have a cascading effect.

Referring to Objects in Other Schemas

The MUSIC user has the SELECT privilege on the COURSE table issued earlier. Observe what occurs when you connect as the MUSIC user in SQL*Plus and attempt to query the table.

```
CONN music/tone
```

```
Connected.
```

```
SELECT description
```

```
FROM course
```

```
FROM course
```

```
*
```

```
ERROR at line 2:
```

```
ORA-00942: table or view does not exist
```

Even though the user MUSIC is allowed to query the COURSE table, MUSIC does not own the COURSE table and must qualify the name of the schema where the object exists. Because the COURSE table exists in the STUDENT schema, you prefix the table name with the schema name.

```
SELECT description
FROM student.course
DESCRIPTION
-----
Technology Concepts
Intro to Information Systems
...
Java Developer III
DB Programming with Java

30 rows selected.
```

672

673

The COURSE table is now qualified with the name of the user who owns the COURSE table (that is, STUDENT). When any query, *Data Manipulation Language* (DML), or *Data Definition Language* (DDL) statement is issued in Oracle, the database assumes that the object being referenced is in the user's own schema, unless it is otherwise qualified.

Private Synonyms

Instead of qualifying the name of an object with the object owner's name, you can use a synonym. A synonym is a way of aliasing an object with another name. You can create private and public synonyms. A private synonym is a synonym in a user's schema; public synonyms are visible to everyone.

The syntax for creating synonyms is as follows.

```
CREATE [OR REPLACE] [PUBLIC]
  SYNONYM [schema.]synonymname
  FOR [schema.]objectname[@dblink]
```

The next CREATE SYNONYM command creates a private synonym called COURSE in the MUSIC schema for the COURSE table located in the STUDENT schema.

```
CREATE SYNONYM course FOR
student.course
```

Synonym created.

If you are not logged in as the MUSIC user but as a user who has rights to create synonyms in another user's schema, such as a DBA, you must prefix the synonym's name with the name of the schema in which the synonym should be created.

```
CREATE SYNONYM music.course FOR
student.course
```

After the synonym is successfully created in the MUSIC schema, you can select from the COURSE table without prefixing the table with the schema name.

```
SELECT description
  FROM course
DESCRIPTION
-----
Technology Concepts
Intro to Information Systems
...
Java Developer III
DB Programming with Java

30 rows selected.
```

Oracle resolves the SELECT statement by looking at the synonym COURSE, which points to the COURSE table located in the STUDENT schema.



Whenever any statement is executed, Oracle looks in the current schema for the object. If there is no object of that name in the current schema, Oracle checks for a public synonym of that name.

When you create a synonym, the validity of the underlying object is not checked; that is, you can create a synonym without the object existing. The synonym is created without error, but you get an error message if you attempt to access the synonym. The following synonym, called SYNONYM_TEST, is based on a nonexistent TEST_ME object, which could be a view, a table, another synonym, or another type of Oracle object.

```
CREATE SYNONYM synonym_test FOR  
test_me
```

Synonym created.

Accessing the synonym results in this message.

```
SQL>SELECT *  
2 FROM synonym_test;  
FROM synonym_test  
*  
ERROR at line 2:  
ORA-00900: anonymous transaction is no longer valid
```

Public Synonyms

All synonyms are private unless the keyword PUBLIC is specified. Public synonyms are visible to all users of the database. However, object privileges are not automatically granted to the underlying objects. You still need to issue grants to either individual users or to PUBLIC by referring to either the public synonym or the underlying object. For the user MUSIC, the following statements create a table, create a public synonym for the table, and grant the SELECT privilege on the table to the user STUDENT.

```
CREATE TABLE instrument
  (instrument_id  NUMBER(10),
   description    VARCHAR2(25))
Table created.
```

```
CREATE PUBLIC SYNONYM instrument FOR instrument
Synonym created.
```

```
GRANT SELECT ON instrument TO student
Grant succeeded.
```

Now the user STUDENT can perform queries against the public synonym or the table INSTRUMENT located in the MUSIC schema. The user STUDENT—or, for that matter, any other user—does not need to prefix the

674

675

INSTRUMENT table with the owner. However, a public synonym does not mean that users other than the user STUDENT have access to the table. If you want every user in the database system to have SELECT privileges, you can grant the SELECT privilege to PUBLIC by using the following command.

```
GRANT SELECT ON instrument TO  
PUBLIC
```



The ability to create public synonyms is typically granted to users with DBA privileges. To complete the exercises in this chapter for public synonyms, have your DBA grant the user STUDENT this privilege or log in as SYSTEM and grant the system privilege CREATE PUBLIC SYNONYM with the following statement:
GRANT CREATE PUBLIC SYNONYM TO student .

Dropping and Renaming Synonyms

You drop synonyms by using the DROP SYNONYM command. The next commands drop the COURSE synonym and the public INSTRUMENT synonym.

```
DROP SYNONYM course
```


Synonym dropped.

```
DROP PUBLIC SYNONYM instrument
```

Synonym dropped.

If a synonym already exists and you want to change its definition, you can use the CREATE OR REPLACE SYNONYM command instead of dropping and re-creating a synonym.

```
CREATE OR REPLACE PUBLIC SYNONYM  
instrument FOR guitar
```

The RENAME command renames a synonym.

```
RENAME instrument TO instrument2
```

Resolving Schema References

Suppose a schema contains a public synonym INSTRUMENT that refers to a table in another user's schema and a table named INSTRUMENT in your own schema. When you issue a query against INSTRUMENT, Oracle resolves the schema reference by referring to the object in your own schema first. If such an object does not exist, Oracle refers to the public synonym.

User-Defined Roles

In addition to using Oracle's predefined system privilege roles (for example, CONNECT, RESOURCE, DBA), you can create user-defined roles to customize a grouping of system and/or object privileges. There may be different types of users for a given system.

675

Sometimes, there are users who only view data, and those users need only SELECT privileges. There are other users who maintain the data, and they typically need a combination of SELECT, INSERT, UPDATE, and DELETE privileges on certain tables and columns. Perhaps programmers need privileges to create procedures, functions, and packages.

676

The syntax to create a role is as follows.

```
CREATE ROLE rolename
```

The following statement creates a role named READ_DATA_ONLY for users who only need to query the data in the STUDENT schema.

```
CREATE ROLE read_data_only  
Role created.
```

The role still does not have any privileges associated with it. The following SELECT statement generates

other statements, granting SELECT privileges on all the STUDENT schema's tables to the new role READ_DATA_ONLY.

```
SELECT 'GRANT SELECT ON ' ||  
table_name ||  
      ' TO read_data_only; '  
FROM user_tables
```

When the statement is executed from a script that in turn executes each resulting statement, the individual commands issued look similar to the following. If you are unsure how dynamic SQL scripts work, refer to [Chapter 14](#), “The Data Dictionary, Scripting, and Reporting.”

```
GRANT SELECT ON COURSE TO  
read_data_only;  
...  
GRANT SELECT ON STUDENT TO  
read_data_only;  
GRANT SELECT ON ZIPCODE TO  
read_data_only;
```

With these individually executed statements, the role READ_DATA_ONLY obtains a collection of privileges. The next step is to grant the READ_DATA_ONLY role to users so these users have the privileges defined by the

role. The following statement grants every user in the database this role by granting the `READ_DATA_ONLY` role to `PUBLIC`.

```
GRANT read_data_only TO PUBLIC
```

Grant succeeded.

Now all users of the database have `SELECT` privileges on all of the `STUDENT` schema's tables. All privileges defined by the role can be revoked in a single statement, such as the following.

```
REVOKE read_data_only FROM PUBLIC
```

Revoke succeeded.

If you want none of the users to have the `SELECT` privilege to the `COURSE` table anymore, you can revoke this privilege from the individual role only, and all users that have been granted this role will no longer have the ability to query the table. This makes the management of privileges fairly easy. If you want to grant the `READ_DATA_ONLY` role only to individual users, such as the `MUSIC` user instead of `PUBLIC`, you can issue a statement such as the following.

```
GRANT read_data_only TO MUSIC
```

676
677

Roles can be granted with the WITH ADMIN option, which allows the user to pass these privileges on to others.

The data dictionary views shown in [Table 15.5](#) list information about roles.

TABLE 15.5 Data Dictionary Views
Related to Roles

DATA DICTIONARY VIEW	PURPOSE
DBA_ROLES	All roles in the database
USER_ROLE_PRIVS	Roles granted to the current user
DBA_ROLE_PRIVS	Roles granted to users and other roles
ROLE_ROLE_PRIVS	Roles granted to roles
ROLE_SYS_PRIVS	System privileges granted to roles
DBA_SYS_PRIVS	System privileges granted to roles and users
ROLE_TAB_PRIVS	Object privileges granted to roles
SESSION_ROLES	Roles a user currently has enabled



The ability to create roles may be performed only by users with DBA privileges or by individual users granted the CREATE ROLE privilege. To complete the exercises in this chapter for user-defined roles, have your DBA grant this privilege to the STUDENT user or log in as SYSTEM and grant this system privilege by executing the statement GRANT CREATE ROLE TO student.

You drop roles by using the DROP ROLE command.

```
DROP ROLE read_data_only  
Role dropped.
```

Profiles

A *profile* is a name for identifying specific resource limits or password features. A user account is always associated with a profile. If at the creation of an account a profile is not specified, the default profile is used. With a profile, you can enforce features such as password expiration settings, maximum idle times (that is, the maximum time without any activity for a session), or the maximum number of concurrent sessions.

677

There are many different profile options; the following syntax lists the most commonly used options.

```
CREATE PROFILE profilename LIMIT
  { {SESSIONS_PER_USER|
    CPU_PER_SESSION|
    CPU_PER_CALL|
    CONNECT_TIME|
    IDLE_TIME}
    {integer|UNLIMITED|DEFAULT}} |
  { {FAILED_LOGIN_ATTEMPTS|
    PASSWORD_LIFE_TIME|
    PASSWORD_REUSE_TIME|
    PASSWORD_REUSE_MAX|
    PASSWORD_LOCK_TIME|
    PASSWORD_GRACE_TIME}
    {expression|UNLIMITED|DEFAULT}} }
```

The next statement creates a profile named **MEDIUM_SECURITY**. With this profile, the password expires after 30 days. If the user logs in with the wrong password, the account is locked after three failed attempts. The password is then locked for one hour (1/24 of a day) unless the DBA unlocks it with the **ALTER USER user_name ACCOUNT UNLOCK** command. The maximum number of concurrent sessions a user may

have is three, and the inactivity time, excluding long-running queries, is 15 minutes.

```
CREATE PROFILE medium_security
LIMIT
  PASSWORD_LIFE_TIME 30
  FAILED_LOGIN_ATTEMPTS 3
  PASSWORD_LOCK_TIME 1/24
  SESSIONS_PER_USER 3
  IDLE_TIME 15
```

Profile created.

When the password expires after the 30 days, the user is prompted to change the password.

You assign a profile to an individual user with the ALTER USER command. The user's resource and password restrictions are then limited within the definition of the profile.

```
ALTER USER music
  PROFILE medium_security
```

User altered.

You can change profiles by using the ALTER PROFILE command, and you can remove them with the DROP PROFILE command. If you drop a profile, any assigned users associated with the profile are automatically assigned the DEFAULT profile.

You can see information about profiles in the data dictionary views `DBA_PROFILES` and `DBA_USERS`. These views are available only if you have the right to see `DBA_` views.

678

679

Additional Security Measures

You can implement stored PL/SQL procedures to encapsulate the security access and business rules to certain transactions. For example, you can create a PL/SQL procedure to update individual employee salaries only during certain hours and within a certain percentage increase range. This way, instead of granting `UPDATE` rights on the `SALARY` column of the `EMPLOYEE` table, you grant the users the right to execute the procedure through which all salary updates must be performed. All the security and logic is enforced and encapsulated within the procedure.

For even finer-grained access control, Oracle provides a feature called the *Virtual Private Database* (VPD) that allows very sophisticated control over many aspects of data manipulation and data access.

Security Implementation

In live production environments, users typically don't log on as owners of the tables they access. Imagine a scenario in which an application user knows the password of the STUDENT account. If the user logs in as the owner of the objects, he or she has the ability to drop tables, update any of the data, or drop any of the indexes in the schema. Needless to say, this situation is a disaster waiting to happen.

A responsible and cautious DBA creates one user account that receives grants for the objects. For example, the DBA may create a STUDENT_USER account to which the application users have access. This account is granted SELECT, INSERT, UPDATE, and DELETE privileges on the various tables.

The DBA then creates synonyms (private or public) so the STUDENT_USER's queries do not need the owner prefix. This STUDENT_USER account cannot drop the STUDENT tables or alter them because the account is not the owner of the table, and the DBA does not grant the system privileges, such as DROP ANY TABLE or ALTER ANY TABLE.

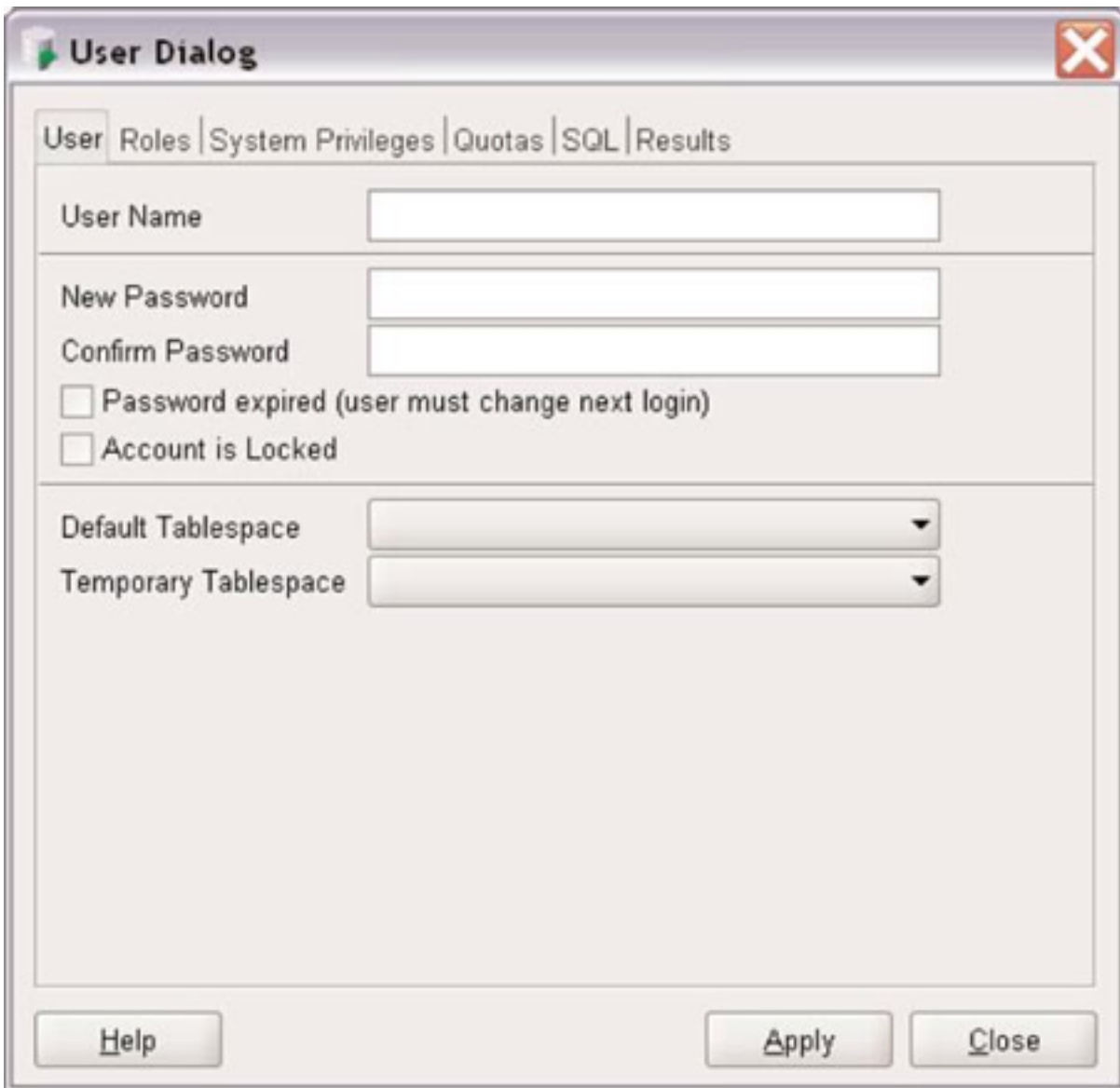
Most individual applications have their own fine-grained security that restricts users to specific screens. There are many ways to implement security, and each individual security implementation depends on the unique requirements of an application. Oracle provides a variety of ways to control and manage the protection of data from unauthorized use.

Creating and Modifying Users and Their Privileges with SQL Developer

SQL Developer has a number of menu options to aid in administering users. While not all syntax options are available, the very commonly used functionality is addressed. Click your Connections node, right-click Other Users, and choose Create User. [Figure 15.1](#) shows the dialog box you use to create a new user. On the User tab, you enter the username and password, and you assign the default and temporary tablespaces from the drop-down lists.

679

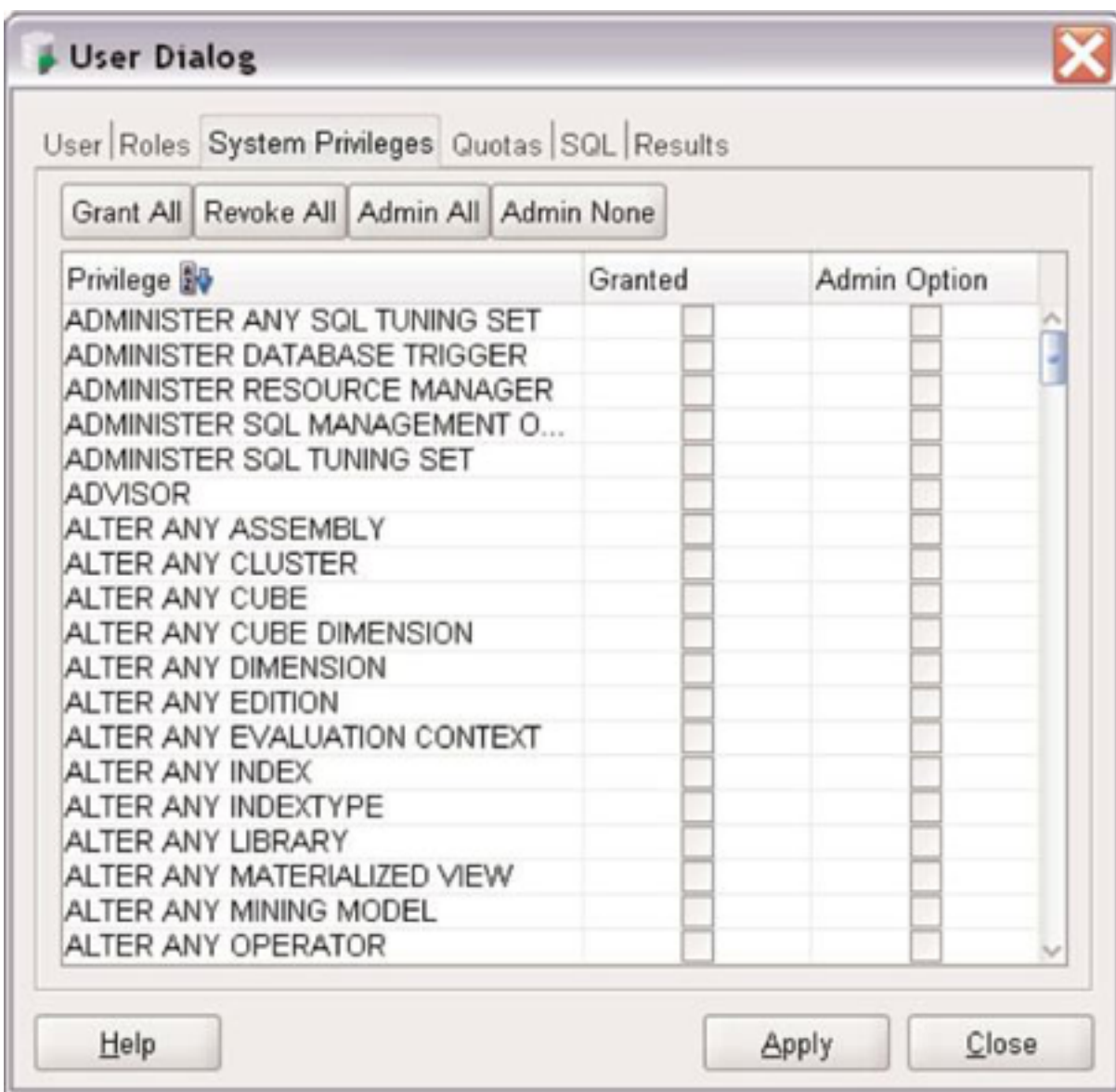
FIGURE 15.1 The User Dialog screen in SQL Developer



The Roles tab lists all the available roles, with check boxes to grant or revoke the roles to the user. Similarly, the System Privileges tab provides a list of all system privileges to choose from (see [Figure 15.2](#)). The Quotas

tab allows you to define access to tablespaces and assign quotas, if any. The SQL tab generates the appropriate SQL statement, and the Results tab shows the execution result after you click the Apply button on the SQL tab.

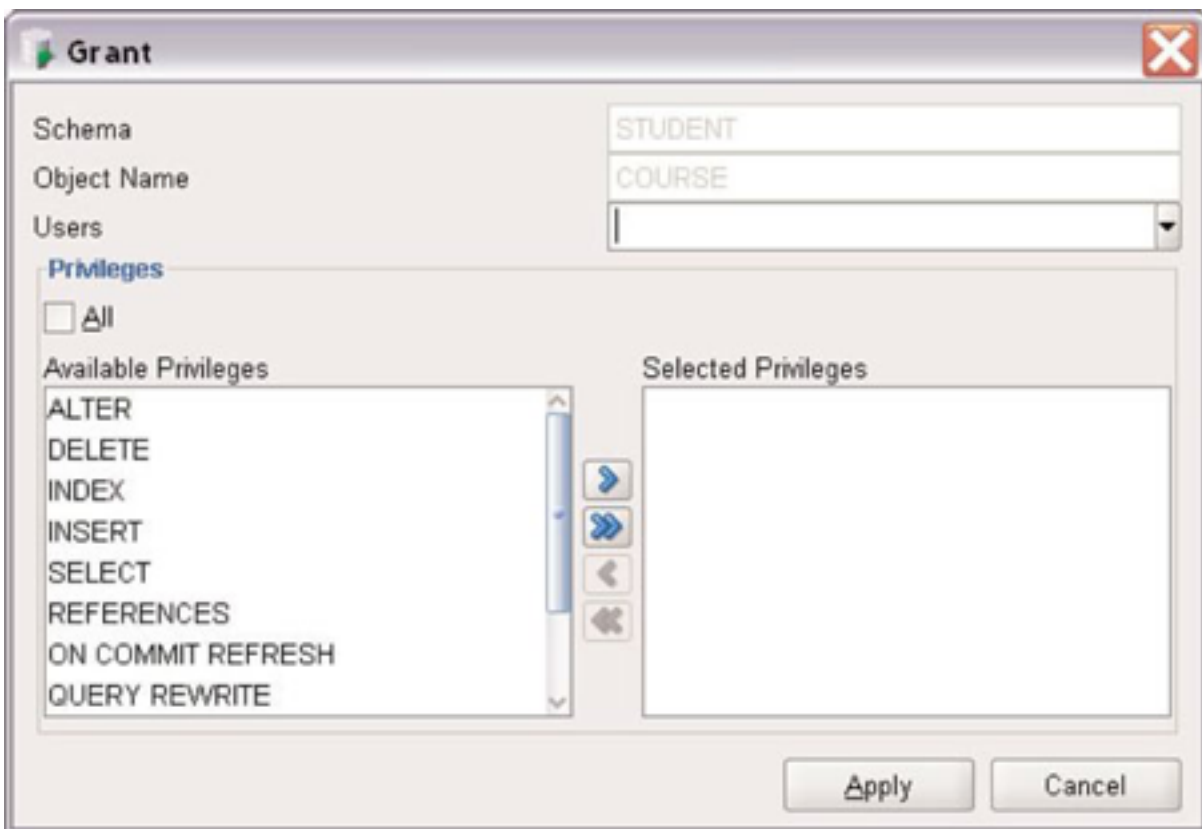
FIGURE 15.2 The System Privileges tab in SQL Developer



680

You may have noticed that object privileges are not part of the User Dialog screen. Instead, to grant or revoke object privileges, right-click the particular object, such as a table, and choose Privileges. If you then choose Grant, you get a screen similar to [Figure 15.3](#). For the given object, you can choose to whom to grant the selected privileges, such as a user or a role.

FIGURE 15.3 The Grant Object privilege screen in SQL Developer



Oracle Accounts

In the exercises so far in this book, you have been using the STUDENT account. During the lab exercises later, you will create different user accounts so that you can better understand the effects of the GRANT and REVOKE commands.

To connect to a different user account in SQL Developer, you need to create a new connection name, using the new user's name and password. In [Chapter 2](#), “SQL: The Basics,” you learned about creating a connection name in SQL Developer.

You can switch between different connections by using the SQL Worksheet icon, which invokes the dialog box shown in [Figure 15.4](#), where you can choose a connection.

In SQL*Plus, you can start another session under a different login name by using the CONNECT command at the SQL*Plus prompt. The CONNECT command can be abbreviated to CONN, followed by the user ID, a forward slash, and the password.

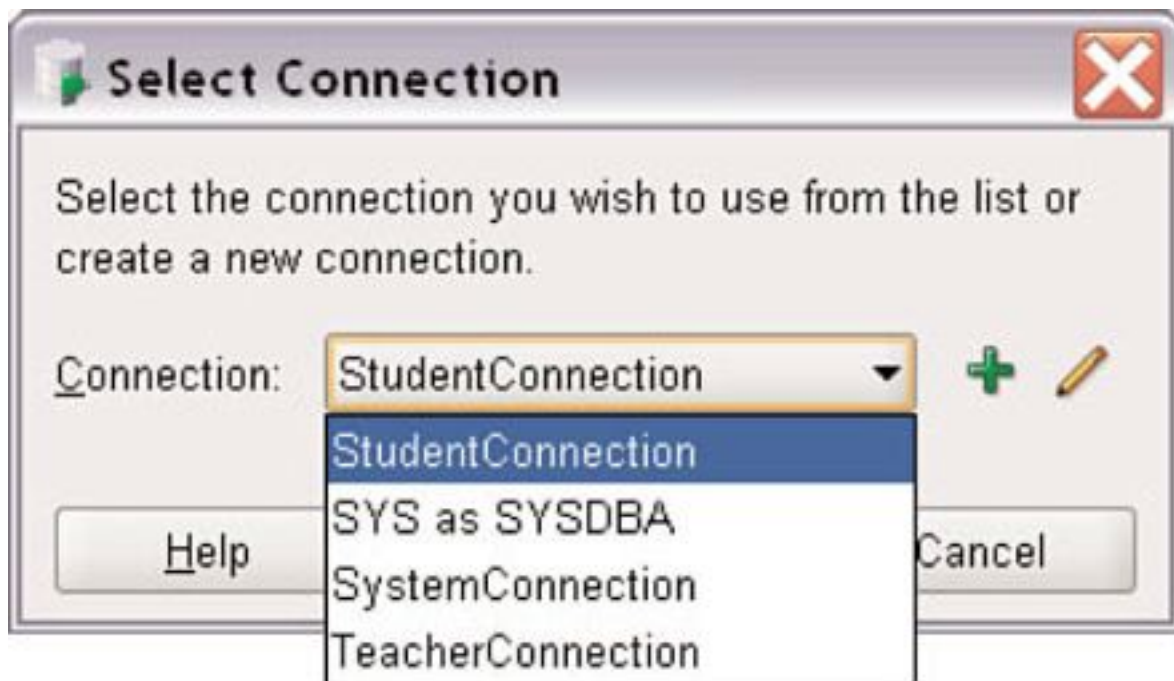
```
SQL> CONN system/manager  
Connected.
```

When you connect as another user while you are running a SQL*Plus session, you are no longer logged in as the previous user. If you prefer, you can just start a new SQL*Plus session to keep both sessions connected.

681

682

FIGURE 15.4 The Select Connection Dialog box in SQL Developer



The following is the syntax for the CONNECT command when using SQL *Plus.

```
CONN[ECT] username/  
password[@connect_identifier] [AS  
{SYSOPER|SYSDBA}]
```


The following is alternative syntax for the CONNECT command.

```
CONN[ECT] / [@connect_identifier] AS  
{SYSOPER|SYSDBA}
```

If you do not supply the password, you are prompted to enter it, as shown in the following example. This may be useful if you do not want to display the password on the screen.

```
SQL> CONN system  
Enter password: *****  
Connected.
```

The following SQL *Plus example shows how you include the host string that identifies the name of the database you want to connect if you are connecting to a database other than the default or local database. In this case, the remote database name is ITCHY, and it is referenced with the @ symbol.

```
CONN system/manager@itchy
```



If during your SQL *Plus session you are unsure which login account you are connected to, issue the SHOW USER command or simply change your SQLPROMPT from SQL> to the current user account with the SET SQLPROMPT _USER> command. The SQLPROMPT command is discussed as part of the predefined SQL*Plus variables in Appendix C, “SQL*Plus Command Reference.”

Connecting with Special Privileges: SYSOPER and SYSDBA

Oracle allows special privileged modes of connection to the database: SYSOPER and SYSDBA. The SYSOPER privilege allows startup and shutdown operations and other basic operational tasks, such as backups, but does not allow the user to look at the user data. The SYSDBA privilege allows you to perform startup and shutdown of the database and to effectively connect as the most privileged user—the SYS user account.

682

You connect with your user schema name and append AS SYSOPER or AS SYSDBA. For example, to connect in SQL*Plus, you can issue the following CONNECT command if the MUSIC user has been granted the SYSDBA privilege.

683

```
CONNECT music AS SYSDBA
```

The user MUSIC is now in a privileged mode of connection that allows the startup or shutdown of the database with the STARTUP or SHUTDOWN command.



When connected as SYSDBA or SYSOPER, you are not connected with the schema associated with your username. Rather, if you connect as SYSOPER, you connect as the owner PUBLIC, and for SYSDBA as the owner SYS, which is the most highly privileged user and owns the data dictionary.

Starting Up and Shutting Down a Database

The following statements show how to shut down a database with the SHUTDOWN command in SQL*Plus.

```
SQL> CONN system AS SYSDBA
Enter password: *****
Connected.

SQL> SHUTDOWN
Database closed.
Database dismounted.
ORACLE instance shut down.
```

To start up the database, you use the STARTUP command.

```
SQL> CONN system AS SYSDBA
```

```
Enter password: *****
```

```
Connected.
```

```
SQL> STARTUP
```

```
ORACLE instance started.
```

```
Total System Global Area  192937984 bytes
```

```
Fixed Size                  769488 bytes
```

```
Variable Size              143573552 bytes
```

```
Database Buffers          25165824 bytes
```

```
Redo Buffers               23429120 bytes
```

```
Database mounted.
```

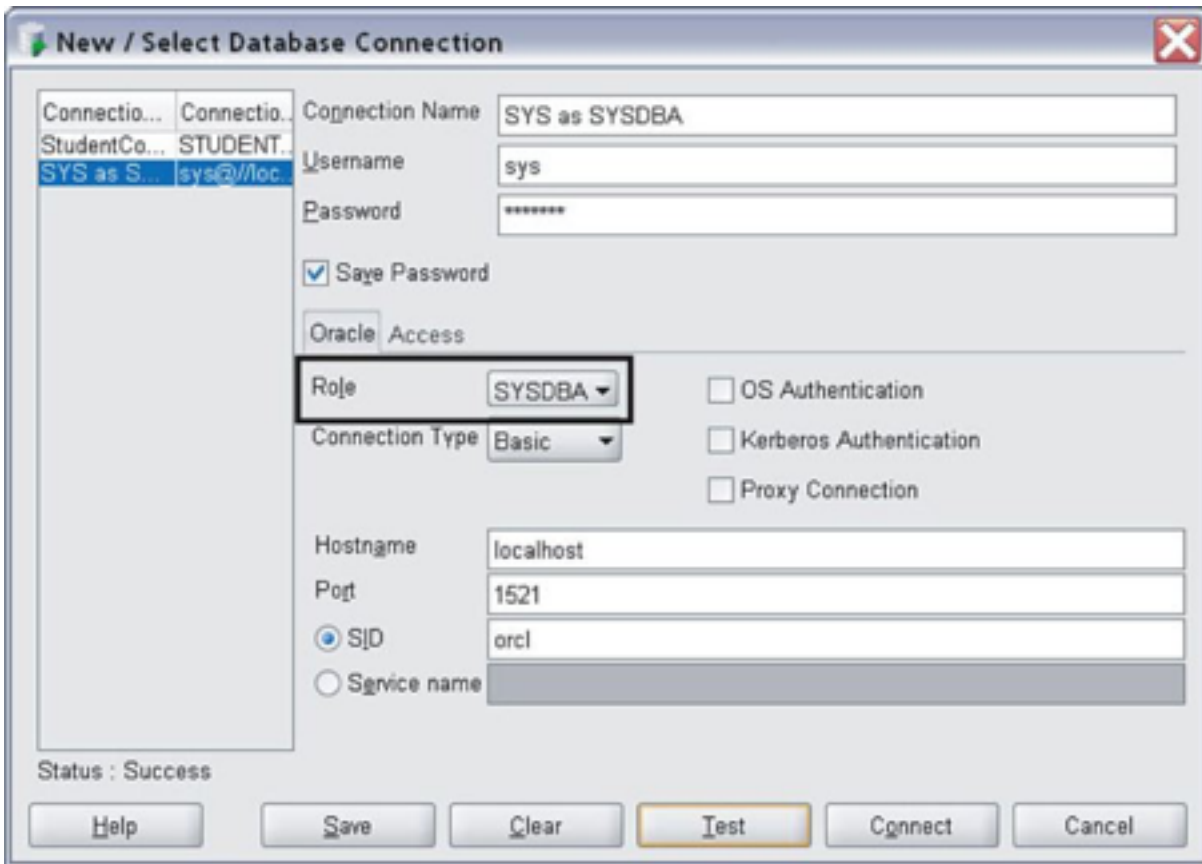
```
Database opened.
```

683

684

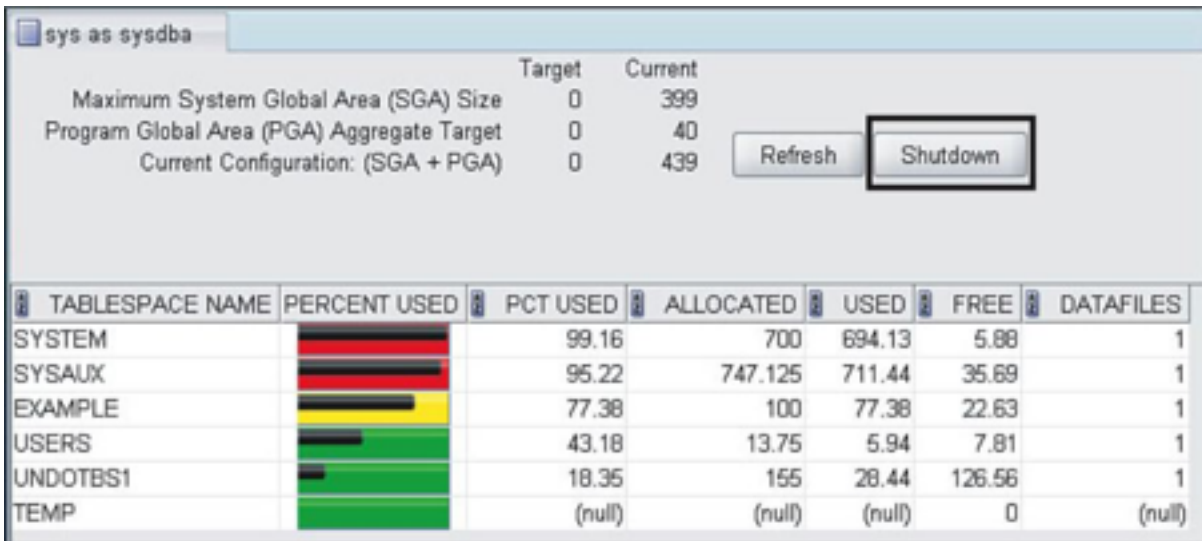
In SQL Developer, you can also connect with the SYSDBA privilege. [Figure 15.5](#) shows an example of connecting as the SYS user with the SYSDBA privilege.

FIGURE 15.5 A SQL Developer database connection



After you have established and tested the connection successfully, you can right-click the connection and choose Manage Database. You then see a screen similar to [Figure 15.6](#), which displays how much memory is allocated to Oracle's SGA and PGA memory areas, as well as a list of tablespaces, with the space availability. Because you are connected as SYSDBA, you see the Shutdown button to the right of the screen.

FIGURE 15.6 The Manage Database screen



684

685

Lab 15.1 Exercises

You can perform a number of the tasks in these exercises by using SQL Developer's various menu choices. Your menu actions result in generated SQL, very much like the SQL commands listed in the exercise answers. If you choose to use SQL Developer, compare the generated SQL to the exercise answers provided here.

- a) Log in as SYSTEM (or any other account that allows you to create a new user) and create a user called TEACHER with the password subject (in lowercase), with the appropriate default and

temporary tablespaces for your database. Using Oracle's predefined roles, grant enough privileges to the new user that the user can create a table. Log in as the new user and create a table called ACCOUNT with these three columns: ACCOUNT_NUM as the primary key column and the columns ACCOUNT_TYPE and ACCOUNT_STATUS. Determine appropriate data types for the columns. Insert a row with the values 1001, Checking, and Active, for these columns.

- b)** While logged in as the new user named TEACHER created in exercise a, execute the following SELECT statements against the data dictionary views. What do these views tell you about the new user?

```
SELECT username, granted_role,  
       admin_option  
   FROM user_role_privs  
SELECT *  
   FROM session_privs
```

- c)** While logged in as the user TEACHER, grant the SELECT privilege for the ACCOUNT table to the STUDENT user and allow the STUDENT

user to grant the same privilege to other users. Then log in as the STUDENT user and execute the following two statements. What do you observe?

```
SELECT *
  FROM teacher.account
INSERT INTO teacher.account
  (account_num, type, status)
VALUES
  (1002, 'Savings', 'Active')
SELECT *
  FROM teacher.account_status
```

- d)** Connect as SYSTEM and change the password for the user TEACHER from subject to class. Log in as TEACHER and revoke the SELECT privileges from the STUDENT user on the ACCOUNT table.
- e)** Execute the following query as the user TEACHER. What purpose does this data dictionary view serve?

```
SELECT username,
  default_tablespace,
  temporary_tablespace
  FROM user_users
```


f) While logged in as the user STUDENT, create a private synonym called COURSE for the COURSE table. Describe your observations.

685

g) Explain the result of the following SELECT statement.

```
SELECT 'CREATE PUBLIC SYNONYM ' ||  
table_name ||  
      ' FOR ' || table_name || ';'   
FROM user_tables
```

686

h) While logged in as the user STUDENT, create a role called STUDENT_ADMIN. Grant INSERT and UPDATE privileges on the COURSE table to the role. Then grant the role to TEACHER.

i) Execute the following SELECT statement and describe the result.

```
SELECT grantee, table_name,  
grantor,  
       privilege, grantable  
FROM user_tab_privs_made
```

LAB 15.1 EXERCISE ANSWERS

The text <default_tablespace> and
<temporary_tablespace> in the following exercise

answers is where the name of the appropriate tablespaces in your database should appear in your answers.

- a) Log in as SYSTEM (or any other account that allows you to create a new user) and create a user called TEACHER with the password subject (in lowercase), with the appropriate default and temporary tablespaces for your database. Using Oracle's predefined roles, grant enough privileges to the new user that the user can create a table. Log in as the new user and create a table called ACCOUNT with these three columns: ACCOUNT_NUM as the primary key column and the columns ACCOUNT_TYPE and ACCOUNT_STATUS. Determine appropriate data types for the columns. Insert a row with the values 1001, Checking, and Active, for these columns.

ANSWER: In SQL*Plus, you can use the CONNECT command to connect as the SYSTEM user. In SQL Developer, you need to create a connection for the SYSTEM account if one does not already exist.

You use the CREATE USER command and the GRANT command to create the new user and grant system privileges to the user.

```
CONN system/manager
```

Connected.

```
CREATE USER teacher IDENTIFIED BY  
subject
```

```
    DEFAULT TABLESPACE  
<default_tablespace>  
    TEMPORARY TABLESPACE  
<temporary_tablespace>
```

User created.

```
GRANT CONNECT, RESOURCE TO  
teacher
```

Grant succeeded.

You use the CONNECT command again to connect as the new user if you use SQL*Plus. In SQL Developer, create a new connection for the TEACHER user and connect via this new connection.

```
CONN teacher/subject
```

Connected.

686

The following CREATE TABLE command creates the new table within the TEACHER schema.

```
CREATE TABLE account
  (account_num NUMBER(15),
   type VARCHAR2(10),
   status VARCHAR2(6),
   CONSTRAINT account_pk PRIMARY
   KEY(account_num))
```

Table created.

```
INSERT INTO account
  (account_num, type, status)
VALUES
  (1001, 'Checking', 'Active')
```

1 row created.

```
COMMIT
```

Commit complete.

- b)** While logged in as the new user named TEACHER created in exercise a, execute the following SELECT statements against the data dictionary views. What do these views tell you about the new user?

```
SELECT username, granted_role,
       admin_option
FROM user_role_privs
```

```
SELECT *
FROM session_privs
```

ANSWER: The query against the `USER_ROLE_PRIVS` view lists what Oracle roles the user `TEACHER` has been granted and whether the user has been granted the administration option on those roles. The query against the `SESSION_PRIVS` view shows the privileges currently available to the user `TEACHER`.

```
SELECT username, granted_role, admin_option
FROM user_role_privs
```

USERNAME	GRANTED_ROLE	ADM
TEACHER	CONNECT	NO
TEACHER	RESOURCE	NO

```
2 rows selected.
```

```
SELECT *
FROM session_privs
PRIVILEGE
```

```
-----
CREATE SESSION
UNLIMITED TABLESPACE
...
CREATE SEQUENCE
```

```
10 rows selected.
```

687

If the following statement had been issued by the SYSTEM account instead, the user TEACHER would be able to grant the same system privileges to another user, becoming the grantor and enabling the TEACHER account to grant these same privileges to another user.

```
GRANT CONNECT, RESOURCE TO
teacher WITH ADMIN OPTION
Grant succeeded.
```

If you subsequently re-execute the query against the USER_ROLE_PRIVS view after issuing the revised GRANT command, you would see YES in the ADMIN_OPTION column.

```
SELECT username, granted_role, admin_option
FROM user_role_privs
```

USERNAME	GRANTED_ROLE	ADM
TEACHER	CONNECT	YES
TEACHER	RESOURCE	YES

```
2 rows selected.
```

These privileges are sufficient to create tables, sequences, and a few other object types, but not user accounts. The ability to create users must be granted individually or via the DBA role to the TEACHER user if you choose to do so. The

following grant needs to be issued by the SYSTEM user.

```
GRANT CREATE USER TO teacher
Grant succeeded.
```

If you want the teacher user to also be able to create views, you need the following system privilege grant.

```
GRANT CREATE VIEW TO teacher
Grant succeeded.
```

- c) While logged in as the user TEACHER, grant the SELECT privilege for the ACCOUNT table to the STUDENT user and allow the STUDENT user to grant the same privilege to other users. Then log in as the STUDENT user and execute the following two statements. What do you observe?

```
SELECT *
  FROM teacher.account
INSERT INTO teacher.account
  (account_num, type, status)
VALUES
  (1002, 'Savings', 'Active')
```

ANSWER: The INSERT statement results in an error due to insufficient privileges.

While logged on as the TEACHER user, issue the GRANT SELECT command on the ACCOUNT table. The WITH GRANT option allows the STUDENT user to pass this privilege on to others.

```
CONN teacher/subject
```

Connected.

```
GRANT SELECT ON account TO
student WITH GRANT OPTION
```

Grant succeeded.

688

The first statement queries the ACCOUNT table in the TEACHER schema without any problems. The SELECT privilege on the ACCOUNT table was granted. The table name must be prefixed with the schema name.

689

```
CONN student/learn
```

Connected.

```
SELECT *
```

```
FROM teacher.account
```

```
ACCOUNT_NUM TYPE STATUS
```

```
-----
```

```
1001 Checking Active
```

```
1 row selected.
```


The second statement attempts to insert a row into the ACCOUNT table. However, the STUDENT user does not have the privilege to perform this action. No INSERT grant on the table was issued to the STUDENT user, and this leads to the insufficient privileges error.

```
INSERT INTO teacher.account
  (account_num, type, status)
VALUES
  (1002, 'Savings', 'Active')
INSERT INTO teacher.account
      *
```

ERROR at line 1:
ORA-01031: insufficient privileges

- d) Connect as SYSTEM and change the password for the user TEACHER from subject to class. Log in as TEACHER and revoke the SELECT privileges from the STUDENT user on the ACCOUNT table.

ANSWER: You use the ALTER USER command to change the password from subject to class. The REVOKE command revokes the SELECT privilege on the ACCOUNT table from the STUDENT user.

```
CONN system/manager
```

Connected.

```
ALTER USER teacher identified by  
class
```

User altered.

```
CONN teacher/class
```

Connected.

```
REVOKE SELECT ON account FROM  
student
```

Revoke succeeded.

If you change the password on the account and saved the password as part of the SQL Developer Connection information, you need to update the password accordingly.

- e) Execute the following query as the user TEACHER. What purpose does this data dictionary view serve?

```
SELECT username,  
default_tablespace,  
temporary_tablespace  
FROM user_users
```

ANSWER: It shows the current user's default and temporary tablespace.

```
USERNAME    DEFAULT_TABLESPACE  TEMPORARY_TABLESPACE  
-----
```

689

690

If the user has any tablespace quotas assigned, you can issue the following query to determine the quota for each tablespace. Obviously, your result may differ from the output listed here.

```
SELECT tablespace_name, bytes/1024/1024 "MB"
FROM user_ts_quotas
```

TABLESPACE_NAME	MB
-----	-----
SYSTEM	.95703125
USERS	13.8125
INDX	.125

3 rows selected.

You assign quotas with the ALTER USER command, as in the following statements. The first assigns 100 M of space on the USERS tablespace to the TEACHER user. Alternatively, the second statement assigns unlimited use of the tablespace USERS to the TEACHER account.

```
ALTER USER teacher QUOTA 100 M ON
users
ALTER USER teacher QUOTA
UNLIMITED ON users
```

A user must have a quota for the tablespace or have been granted the UNLIMITED TABLESPACE system privilege (for example,

through the RESOURCE role) to be able to create indexes and tables.

- f) While logged in as the user STUDENT, create a private synonym called COURSE for the COURSE table. Describe your observations.

ANSWER: Two objects with the same name cannot exist in the same schema.

```
CREATE SYNONYM course FOR course
CREATE SYNONYM course FOR
course
*
```

ERROR at line 1:
ORA-01471: cannot create a
synonym with same name as object

It is not necessary to create private synonyms for objects you already own. It is possible to do so, but the synonym must have a different name from the underlying object. Within one schema, all object names must be unique, regardless of the type of object.



Public synonyms are not owned by the user who creates them, so there is no conflict between the public synonym name and the name of the object on which it is based.

690

- g) Explain the result of the following SELECT statement.

```
SELECT 'CREATE PUBLIC SYNONYM ' ||  
table_name ||  
      ' FOR ' || table_name || ';'   
FROM user_tables
```

691

ANSWER: The SELECT statement generates other SELECT statements dynamically. Each statement generated creates a public synonym for each table owned by the current user.

When you create public synonyms for other users to see your objects, you typically do it for many objects in your schema. Using a SELECT statement to generate other statements is the fastest way to do this.

- h) While logged in as the user STUDENT, create a role called STUDENT_ADMIN. Grant INSERT and UPDATE privileges on the COURSE table to the role. Then grant the role to TEACHER.

ANSWER: First, you use the CREATE ROLE command to create the role. Then you issue the GRANT command to grant INSERT and UPDATE privileges, separated by commas, to the role. Another GRANT statement grants the role to the user TEACHER.

```
CREATE ROLE student_admin
```

Role created.

```
GRANT INSERT, UPDATE ON course TO  
student_admin
```

Grant succeeded.

```
GRANT student_admin TO TEACHER
```

Grant succeeded.

You can use WITH ADMIN OPTION to pass on the ability to grant the privileges being granted. The following statement is the same as the previous GRANT statement except that it also gives the TEACHER user the ability to pass on the privileges that are being granted.

```
GRANT student_admin TO teacher  
WITH ADMIN OPTION
```

Grant succeeded.

Now the user TEACHER can pass the same set of privileges to other users.

- i) Execute the following SELECT statement and describe the result.

```
SELECT grantee, table_name,  
       grantor,  
       privilege, grantable  
FROM user_tab_privs_made
```

ANSWER: The result shows the details of all grants made on tables by the STUDENT user: the recipient of the grant (the grantee); the table on which the grant was based; the grantor, or the user who granted the privilege; the privilege granted on the table; and whether the privilege can be granted to other users.

The results vary, depending on the privileges you have granted and have been granted by other users.

GRANTEE	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE
STUDENT_ADMIN	COURSE	STUDENT	INSERT	NO
STUDENT_ADMIN	COURSE	STUDENT	UPDATE	NO

2 rows selected.

You can see that the STUDENT_ADMIN role is the grantee of INSERT and UPDATE privileges on the COURSE table, and the STUDENT user is the grantor.

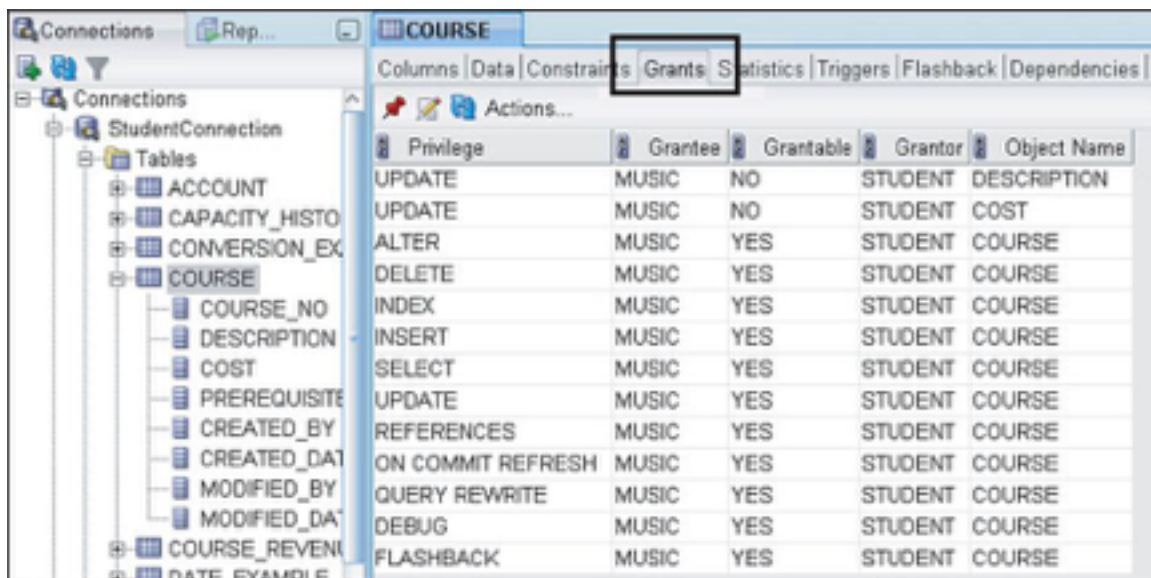
691

692

You can query the DICT data dictionary view to list several other data dictionary views containing information about the roles created and privileges granted in a system or refer to [Tables 15.4](#) and [15.5](#).

You can easily view any object grants issued by reviewing the Grants tab for a chosen object. [Figure 15.7](#) shows the Grants tab in the COURSE table, with details on the grantee, whether the privilege is grantable, and so on.

FIGURE 15.7 The Grants tab for an object in SQL Developer



Privilege	Grantee	Grantable	Grantor	Object Name
UPDATE	MUSIC	NO	STUDENT	DESCRIPTION
UPDATE	MUSIC	NO	STUDENT	COST
ALTER	MUSIC	YES	STUDENT	COURSE
DELETE	MUSIC	YES	STUDENT	COURSE
INDEX	MUSIC	YES	STUDENT	COURSE
INSERT	MUSIC	YES	STUDENT	COURSE
SELECT	MUSIC	YES	STUDENT	COURSE
UPDATE	MUSIC	YES	STUDENT	COURSE
REFERENCES	MUSIC	YES	STUDENT	COURSE
ON COMMIT REFRESH	MUSIC	YES	STUDENT	COURSE
QUERY REWRITE	MUSIC	YES	STUDENT	COURSE
DEBUG	MUSIC	YES	STUDENT	COURSE
FLASHBACK	MUSIC	YES	STUDENT	COURSE

692

Lab 15.1 Quiz

In order to test your progress, you should be able to answer the following questions.

1) A user's objects must be dropped in a separate statement before the user can be dropped.

_____ **a)** True

_____ **b)** False

2) The SQL*Plus CONNECT command is not the same as the CONNECT role.

_____ **a)** True

_____ **b)** False

3) The following statement contains an error.

```
REVOKE resource, SELECT ON course  
FROM music
```

_____ **a)** True

_____ **b)** False

4) System privileges cannot be granted through a role.

_____ **a)** True

_____ **b)** False

5) Dropping a role drops the underlying object on which the role's privileges are based.

_____ **a)** True

_____ **b)** False

6) Grants can be given to users, roles, or PUBLIC.

_____ **a)** True

_____ **b)** False

7) After issuing a DCL command, you must execute the COMMIT command to make a change permanent.

_____ **a)** True

_____ **b)** False

8) The SYSTEM Oracle account is a highly privileged account.

_____ **a)** True

_____ **b)** False

ANSWERS APPEAR IN APPENDIX A.

693

Workshop

The projects in this section are meant to prompt you to utilize all the skills you have acquired throughout this chapter. The answers to these projects can be found at the companion Web site to this book, located at www.oraclesqlbyexample.com.

To complete the following exercises, create a new user called SCHOOL with the password program and grant CONNECT and RESOURCE roles to this user. Then log in as the STUDENT user.

- 1) Create two roles: REGISTRAR and INSTRUCTOR.
- 2) Create a view called CURRENT_REGS that reflects all students who registered on January 25, 2007. Grant the SELECT privilege on the new view to the REGISTRAR role.
- 3) Create a view called ROSTER that reflects all students taught by the instructor Marilyn Frantzen. Grant the SELECT privilege on the new view to the INSTRUCTOR role.
- 4) Grant the REGISTRAR and INSTRUCTOR roles to the new user called SCHOOL.

5) Log in as the user SCHOOL and select from the two previously created views.