CHAPTER 11

Managing Users and Security

In This Chapter

Creating and managing users

Granting roles and privileges to users

Creating roles

Creating and assigning user profiles

Becoming a user

Viewing users, security, and the data dictionary

For many systems, the DBA is in charge of security. Security often becomes a time-consuming task that needs regular attention. Oracle's Security Manager, part of Enterprise Manager's DBA Management Pack, makes your tasks more intuitive and helps you manage users, passwords, and database privileges more efficiently and accurately.

This chapter explores how you can use Security Manager to create users and roles, assign privileges, and control passwords. It also shows you how to perform these tasks manually, executing SQL statements from the SQL*Plus tool. While Security Manager provides a GUI interface that substitutes for handwritten SQL code, underneath that interface, SQL statements are still being generated and executed. Security Manager will let you view those statements. Viewing the SQL statements generated by Security Manager is one way to reinforce your knowledge of SQL.

Creating and Managing Users

Often, one of your first tasks involves defining new users. Later, when a definitive security plan evolves, you can establish roles with specialized sets of privileges to enforce security. Typically, each end user becomes a member of at least one role that you create.

Creating a new user

The SQL command for creating a user is, aptly enough, CREATE USER. Before using this command to create a new user, keep the following in mind:

- ✦ A value for the user's initial password
- ✦ A default tablespace in which you want to store objects owned by the user
- ✦ The temporary tablespace to be used when the user executes queries requiring sorts
- ✦ The amount of disk space that you wish to allow the user to use in the default tablespace

Once you have these items in mind, you can create a new user by issuing a SQL statement like this one:

```
CREATE USER joseph IDENTIFIED BY big_cave
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp
    QUOTA UNLIMITED ON users;
```

It is not necessary to give quota on the temporary tablespace.

This statement creates a new user named JOSEPH and assigns him an initial password of BIG_CAVE. Joseph's default tablespace will be the USERS tablespace, and his temporary tablespace will be TEMP. Joseph has been given unlimited quota on the USERS tablespace, so he can use any amount of disk space in that tablespace.

Default Tablespaces

When a user creates an object such as a table or an index, that object must be stored somewhere. In Oracle, you control an object's storage by assigning it to a tablespace. If you don't specify a tablespace when you create an object, Oracle will pick one for you. The tablespace that Oracle picks will be the one specified as your default tablespace when your user ID was created.

The benefit of being able to specify a default tablespace for a user is that it allows users who aren't very sophisticated or knowledgeable about Oracle to create tables, indexes, and so forth, without having to worry about where they are stored. As the DBA, you can still exert some control over where objects are placed because you control the default tablespace assignments.

You should assign every user a default tablespace. If you don't specify a default tablespace when creating a user, then the SYSTEM tablespace is used as the default, which isn't good. The SYSTEM tablespace contains the data dictionary and is

heavily used by Oracle. Placing user objects in the same tablespace can lead to performance degradation because of disk contention. Even if you don't ever expect a user to create any objects, specify a default tablespace anyway. That way, you won't be surprised later.

Temporary Tablespaces

Several types of SQL queries require data to be sorted. The obvious examples are those queries containing `ORDER BY` clauses, but Oracle can also perform sorts for `GROUP BY` queries, `DISTINCT` queries, and for certain types of joins. Some sorts can be done in memory, but if the amount of data to be sorted is large enough, the sorting process requires that some of it be temporarily written to disk.

When Oracle needs temporary storage, it will look to see if the user has been assigned a temporary tablespace and will perform the sort there. If that tablespace has been created specifically as a temporary tablespace, the result will be better performance, because Oracle's sorting routines have been optimized to take advantage of temporary tablespace features.

Quotas

A *quota* is a limit on the amount of disk space that one user is allowed to use. When you create a user, you have the option of granting the user permission to use some disk space in your database. You do this by placing one or more `QUOTA` clauses in the `CREATE USER` command. The earlier example used the keyword `UNLIMITED` to allow the user `JOSEPH` to use any amount of disk space in the `USERS` and `TEMP` tablespaces.

You aren't limited to specifying quota for just the default and temporary tablespaces. You can assign quota on any tablespace in the database. If you were to later decide that `JOSEPH` needed to create objects in the `SEAPARK_DATA` and `SEAPARK_INDEXES` tablespaces, you could issue this command:

```
ALTER USER joseph
    QUOTA 10M ON SEAPARK_DATA
    QUOTA 5M ON SEAPARK_INDEXES;
```

Here, Joseph has been granted the authority to use up to 10MB of space in the `SEAPARK_DATA` tablespace, and up to 5MB of space in the `SEAPARK_INDEXES` tablespace. The M following the numbers in the command stands for MB. You can also use a suffix of K (for KB). If you give a user quota on a tablespace, and later decide that you don't want the user to have any quota on that tablespace, you can set the user's quota for that tablespace to zero.

Choosing Names for Users and Passwords

As a DBA, you have the responsibility of creating every new username for your Oracle8i database. When setting up a name for a new user, you must follow the same rules that apply for naming any Oracle object:

- ✦ Names may be up to 30 characters long.
- ✦ Names must begin with a letter.
- ✦ After the first letter, names may consist of any combination of letters (A-Z), digits (0-9), underscores (_), pound signs (#), and dollar signs ($).
- ✦ Names are not case sensitive unless they are enclosed within double quotes.
- ✦ Names within double quotes may contain any combination of characters, in any order, but may not contain embedded quotes.

Using Security Manager to create a user

Security Manager makes quick work of creating a user. Its GUI interface is easy to follow and does not require you to remember any SQL syntax. To use Security Manager to create a user, follow these steps:

1. Start up Security Manager and log on to the database.

   Figure 11-1 shows the opening window of Security Manager.

2. Right-click the Users folder, and select Create to create a new user.

   This will open the Create User dialog box, which is shown in Figure 11-2.

You can also click the Create button on the toolbar or select Create from the Object menu to begin creating a user. With either of these two methods, you will need to proceed through another dialog box where you choose the type of object (user, role, or profile) that you are creating.

3. Type a new username.
4. Select a profile set from the pull-down list.

   Refer to the section "Creating and Assigning User Profiles" later in this chapter to learn how to create new profiles.

Figure 11-1: Security Manager's opening window

Figure 11-2: The Create User dialog box

5. Select the authentication type.

   Three types of authentication exist:

   - **Global.** A username can be defined as unique across multiple databases by selecting global authentication.
   - **External.** Oracle validates the user through the operating system. In these cases, you append a common prefix to the user's operating system name to create the Oracle username. The default prefix is OPS$. If a user logs on to the operating system as JOSEPH, then the user's Oracle username is OPS$JOSEPH. The user doesn't enter a password when logging on to Oracle with external authentication.
   - **Password.** The user must enter a password when logging on to the database. This is the traditional method. Use it if you are uncertain which choice to make.

6. If the user type is Password, type the password in both the Enter Password box and the Confirm Password box.

   To prevent other people from seeing it, the password will appear on the screen as a string of asterisks (*).

   Oracle enables you to require the user to enter a new password the first time the user logs on. Select the Expire Password Now check box to use this feature.

7. Select a default tablespace.

   This is the tablespace in which Oracle puts the new user's tables if the user creates tables without explicitly assigning them to a different tablespace.

8. Click the Temporary drop-down list arrow to select a temporary tablespace.

   Temporary tablespaces are normally named with `TEMP` as part of the name. The default database that you get with Oracle8i uses `TEMP` for the name of the temporary tablespace.

9. Click the Quota tab (optional).

   Figure 11-3 shows the Quota tab. Here you can assign limits to the amount of space a user is allowed to use in each of the tablespaces in the database. To assign a limit, select the tablespace, click the Value option, type a number, and then select either K Bytes or M Bytes from the drop-down list. Do this for each tablespace that you want the user to use, except for the temporary tablespace. You do not need to assign quota for temporary tablespaces.

10. Click the Create button to finish.

    Oracle creates the new username and displays a message telling you that the creation was successful.

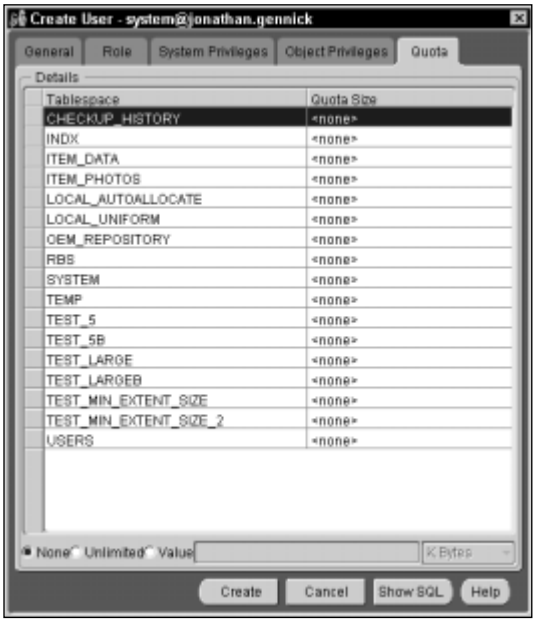

Figure 11-3: The Quota tab in the Create User dialog box

There's an easy way to use an existing user as a template when creating a new user. Right-click the template user in the user list, and then select Create Like from the pop-up menu. Fill in a new username and password. Make any other changes that you would like, and then click the Create button to create the new user.

Changing a user's password

Oracle does not contain a utility to display a user's password — it always appears as asterisks or in an encrypted form. If a user forgets his or her password, you must assign a new one. You can either use the `ALTER USER` command to change a password, or you can do it from Security Manager.

Changing a Password Using the ALTER USER Command

The following example shows you how to use the `ALTER USER` command to change a user's password. It changes Joseph's password to `BROWN_BAG`.

```
ALTER USER joseph IDENTIFIED BY brown_bag;
```

As you can see, this command is simple. You can issue it from either SQL*Plus or SQLPlus Worksheet.

Using Security Manager to Change a Password

To change a user's password from Security Manager, follow these steps:

1. Start Security Manager, and log on to the database.
2. Double-click the User folder in the left pane.

   This brings up a list of users on the left side of the screen.
3. Click the user whose password you are changing.

   The General tab of the user's Profile appears in the right pane, as shown in Figure 11-4.

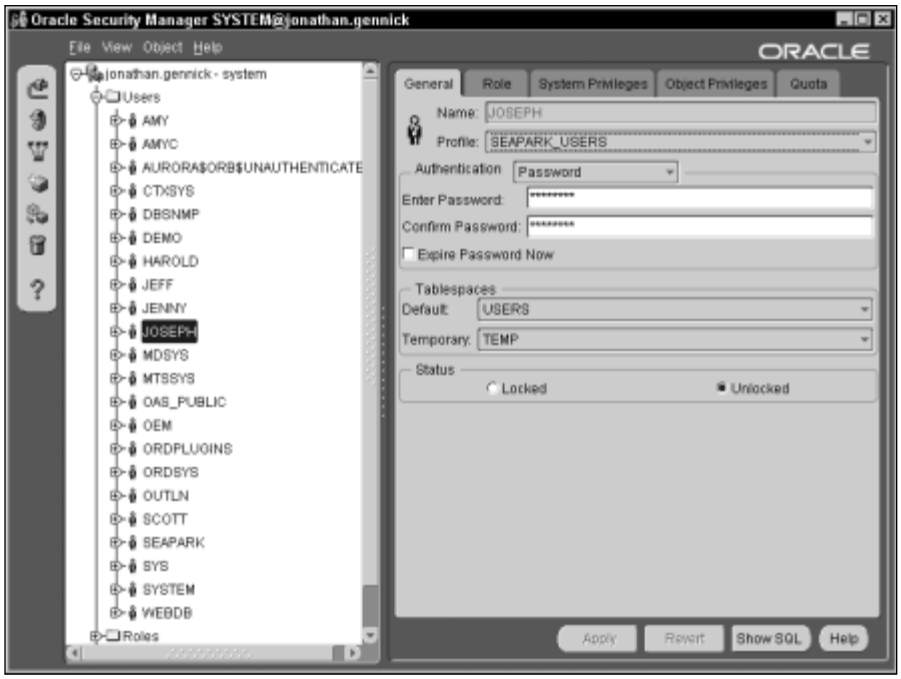

Figure 11-4: Editing a user's profile

4. Type the new password in the Enter Password box and again in the Confirm Password box.
5. Click the Apply button to complete the job.

   Security Manager executes the SQL statements necessary to apply the change and returns you to the main window.

Changing user settings

You can change the default tablespaces, account lock, profile, or quotas assigned to any user. Use the ALTER USER command to perform these tasks, or if you prefer, you can do them from Security Manager.

The account lock/unlock feature enables you to lock out users from accessing a database.

To make any of these changes from Security Manager, follow the procedure described earlier for making a password change. Once you have a user record selected for editing, you can make any changes you like to that record. The following sections show you how to make these changes from SQL*Plus.

Changing Tablespaces

You can change the default tablespace for a user by issuing a command like this:

```
ALTER USER username
    DEFAULT TABLESPACE tablespace_name;
```

You can change a user's temporary tablespace using the same method; just use the keyword TEMPORARY in place of DEFAULT.

If you change a user's default tablespace, don't forget to assign quota on that tablespace.

Locking Out a User

You can temporarily lock a user's account and prevent him or her from accessing the database by using this command:

```
ALTER USER username ACCOUNT LOCK;
```

You can re-enable the user's access by unlocking the account, using this command:

```
ALTER USER username ACCOUNT UNLOCK;
```

This account lock feature was introduced in Oracle release 8.0, and it provides a convenient method for temporarily disabling a user without having to drop the user.

Changing a User's Profile

To change a user's profile, use the ALTER USER command as follows:

```
ALTER USER username PROFILE profile_name;
```

Replace *profile_name* with the name of any valid profile defined for the database.

Changing a User's Quota

To change a user's disk quota on a tablespace, issue an ALTER USER command like this:

```
ALTER USER username
    QUOTA {UNLIMITED|VALUE[M|K]} ON tablespace_name;
```

For example, to reduce Joseph's quota on the USERS tablespace from 10MB to 5MB, issue the following command:

```
ALTER USER username
    QUOTA 5M ON users;
```

You can use as many QUOTA clauses in the command as necessary. Reducing a user's quota doesn't affect data that the user already has stored. Had Joseph already been using 8MB of data, for example, in the USERS tablespace, reducing his quota to 5MB would not result in any data being deleted.

Dropping a user

There are two ways to delete a user from the database. One way is to issue the DROP USER command from SQL*Plus or SQLPlus Worksheet. The other way is through Server Manager. If a user owns tables, indexes, or other objects, you normally need to delete those objects before you can drop the user. Using the CASCADE option allows you to drop a user and his or her data in one shot.

Using the DROP USER Command

Use the DROP USER command to delete a user from the database. The syntax is simple. To drop the user named JOSEPH, you issue the following command:

```
DROP USER joseph;
```

This form of the command requires that there are no tables, indexes, or other objects owned by the user.

Dropping a User and the User's Data

To drop a user, together with all tables and other objects owned by that user, add the CASCADE option to the DROP USER command. For example:

```
DROP USER joseph CASCADE;
```

Be careful with this command. You can easily delete a lot of data by mistake.

Using Security Manager to Drop a User

In addition to creating users, Security Manger allows you to delete them. To use Security Manager to delete (or drop) a user, follow these steps:

1. Run Security Manager, and log on to the database.
2. Expand the Users folder. You can do this by double-clicking the folder or by clicking the plus (+) sign to the left of the folder.
3. Find the user that you want to delete. Right-click that user, and select the Remove option from the pop-up menu.
4. You will be asked to confirm your intention to drop the user, as shown in Figure 11-5.

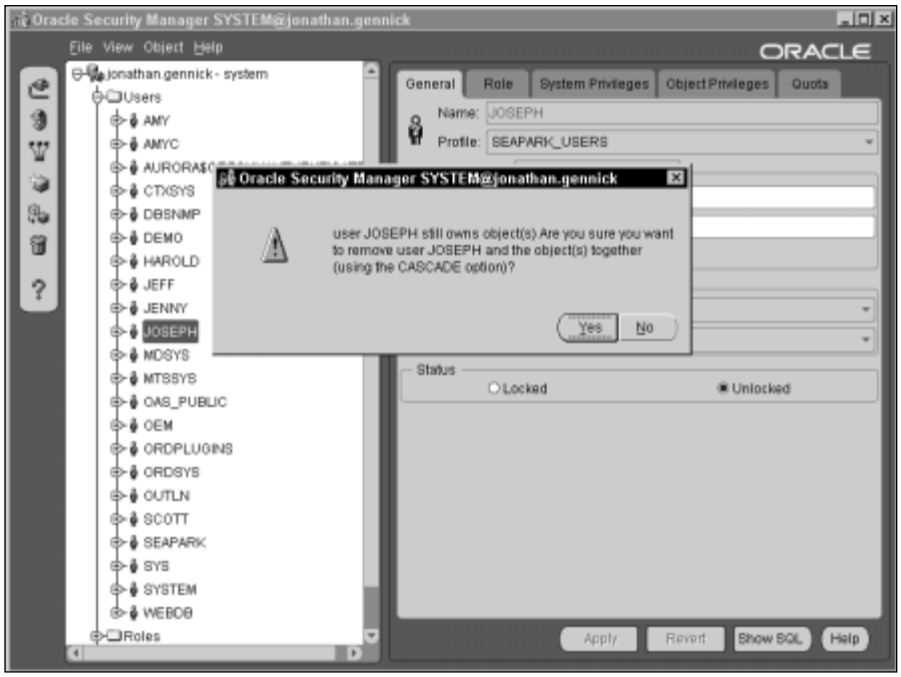

Figure 11-5: Using Security Manager to delete a user

Unlike issuing the `DROP USER` command, you don't need to decide ahead of time whether you need to use the `CASCADE` option. If you are deleting some objects that the user owns, Security Manager will warn you and then apply the `CASCADE` option automatically.

Granting Roles and Privileges to Users

For a user to do anything to a database, you need to grant that user one or more system privileges. A *system privilege* is a privilege defined by Oracle that allows a user to perform a certain task, such as creating a table. Oracle has defined a rich set of privileges that you can grant or not grant to closely control what a user is allowed to do. Even just logging on to a database requires a privilege — the `CREATE SESSION` privilege.

Privileges are frequently bundled together into roles. You'll learn more about roles, including how to create them, later in this chapter. One of the most commonly granted roles is the `CONNECT` role, a default role that you get when you create a database. It conveys the privileges required to log on to the database and to create common objects such as views, tables, indexes, and so forth.

To see a list of privileges conveyed by a role such as `CONNECT`, navigate to that role in Security Manager and click it. The list of privileges granted to the role will appear in the right pane.

Oracle provides two types of privileges: system privileges and object privileges. System privileges give you the ability to do various tasks in an Oracle database. System privileges are not used to provide access to the tables and views within that database. Object privileges are used to specify the type of access to the data a user can have in various objects such as tables, views, and sequences.

Granting system privileges and roles

You can use the `GRANT` command to assign a system privilege or a role to a user. Similarly, you can use the `REVOKE` command to remove a system privilege or a role from a user. You can issue the `GRANT` and `REVOKE` commands from a command-line utility such as SQL*Plus, or you can use Security Manager's GUI interface to manage privileges.

The GRANT Command

To grant a system privilege or a role to a user, use the `GRANT` command as follows:

```
GRANT {privilege|role}[,{privilege|role}...]
TO {username|rolename};
```

You can list as many privileges and roles in the command as you need. The `CREATE SESSION` privilege allows a user to log on to a database. The `CREATE TABLE` command allows a user to create tables and indexes. You can grant both to the user named `JOSEPH` using this command:

```
GRANT CREATE SESSION, CREATE TABLE TO joseph;
```

You grant a role in the same way that you grant a privilege. To grant Joseph the CONNECT role, you can issue this command:

```
GRANT CONNECT TO joseph;
```

You can grant a privilege or a role to all users in the database by using the keyword PUBLIC in place of a username. For example, the following GRANT allows any user to use any amount of disk space within the database:

```
GRANT UNLIMITED TABLESPACE TO PUBLIC;
```

Granting UNLIMITED TABLESPACE to a user has the same effect as if you had given that user an unlimited quota on each tablespace within the database.

Commonly Granted Privileges

The list of privileges that Oracle supports is quite long. Some are used more frequently than others. Table 11-1 contains a list of some of the more commonly used privileges.

Table 11-1
Commonly Granted Privileges and Roles

Privilege/Role	*Description*
CREATE SESSION	Allows a user to log on to the database.
CONNECT	Allows you to log on and create the most-used objects. The CONNECT privilege is a predefined role that conveys a bundle of useful system privileges.
RESOURCE	Conveys UNLIMITED TABLESPACE and is similar to CONNECT. The list of objects that you can create is longer, but RESOURCE does not convey the CREATE SESSION privilege. When you grant RESOURCE to a user, the UNLIMITED TABLESPACE privilege is automatically granted as well.
CREATE TABLE	Allows the user to create tables.
CREATE VIEW	Allows the user to create views.
CREATE SEQUENCE	Allows the user to create sequences.
CREATE PROCEDURE	Allows the user to create procedures, functions, and packages.
CREATE TRIGGER	Allows the user to create triggers on tables owned by that user.
CREATE SYNONYM	Allows the user to create private synonyms.

For a complete list of Oracle database privileges, see the entry for the GRANT command in Appendix A, "SQL Statement Reference."

The ADMIN Option

By default, only database administrators can grant roles or privileges to other users. If you happen to have a specific role or privilege that you want someone else to manage for you, you can grant it to him or her using the ADMIN option. For example:

```
GRANT CREATE TABLE TO joseph WITH ADMIN OPTION;
```

The ADMIN option allows Joseph to further grant CREATE TABLE to other users besides himself. The ADMIN option can be very useful if you have created a set of roles in support of a specific application. You can grant those roles, with the ADMIN option, to an application administrator, who can then shoulder the burden of properly granting those roles to application users.

Revoking System Privileges and Roles

You can use the REVOKE command to remove a role or a privilege from a user. The syntax looks like this:

```
REVOKE {privilege|role}[,{privilege|role}...]
FROM {username|rolename}
```

For example, the following command revokes CREATE TABLE from the user named JOSEPH:

```
REVOKE CREATE TABLE FROM joseph;
```

If you have granted a privilege to PUBLIC, you can revoke it by using PUBLIC in place of the username. For example:

```
REVOKE UNLIMITED TABLESPACE FROM PUBLIC;
```

As with the GRANT statement, you may revoke any number of roles and privileges using one command.

Granting object privileges

Object privileges allow a user access to the data within a table, view, sequence, procedure, or other object. The five commonly used object privileges are the following:

SELECT	Allows a user to retrieve data from a table, sequence, or view
UPDATE	Allows a user to change data in a table or view

DELETE	Allows a user to delete data from a table or view
INSERT	Allows a user to add new rows to a table or view
EXECUTE	Allows a user to execute a stored procedure, function, or package

You use the GRANT and REVOKE commands to manage object privileges, but the syntax is slightly different than that used when dealing with system privileges and roles.

Object privileges must be granted and revoked by the owner of the object in question. Even database administrators cannot grant object privileges for objects owned by another user.

Granting an Object Privilege

The format of the GRANT command you can use when granting object privileges looks like this:

```
GRANT {privilege_list|ALL} [(column_list)]
ON object TO {user|role|PUBLIC}
[WITH GRANT OPTION];
```

The following list describes the parameters of this command:

privilege_list	This is a comma-delimited list of object privileges.
ALL	This grants all privileges relevant to the object in question.
column_list	This is a comma-delimited list of columns, and is only applicable when granting INSERT, UPDATE, or REFERENCES on a table or a view.
object	This is an object name. This may optionally be qualified with a schema name, using the standard dot notation.
user	This is the user to whom you want to grant the privilege(s).
role	This is the role to which you want to grant the privilege(s).
PUBLIC	This is used to grant one or more privileges to all users.
WITH GRANT OPTION	This allows the user to further grant the privilege to another user.

Only an object owner can grant access to its objects. Even database administrators, who may be able to read that data because of system privileges that they hold, are not allowed to grant access to objects owned by another user.

By using WITH GRANT OPTION, an object owner can grant a specific object privilege to another user and allow that user to further grant it to others.

As an example, suppose that you want to grant the user named HAROLD the ability to query the checkup history of animals in Seapark, and that you also want HAROLD to be able to insert new checkup records. To do all this, you need to log on as the SEAPARK user and issue the following commands:

```
GRANT SELECT ON AQUATIC_ANIMAL TO harold;
GRANT SELECT ON CHECKUP TO harold;
GRANT SELECT, INSERT ON CHECKUP_HISTORY TO harold;
```

The SELECT privileges are needed on the first two tables, because HAROLD will need to query for the animal names and for the checkup types. The INSERT privilege is needed on CHECKUP_HISTORY, because each checkup is recorded by adding a record to that table.

Revoking an Object Privilege

The format of the REVOKE command to remove object privileges from a user looks like this:

```
REVOKE {privilege_list|ALL} ON object
FROM {user|role|PUBLIC} [CASCADE CONSTRAINTS] [FORCE];
```

With two exceptions, the syntax elements in the REVOKE command have the same meanings as in the GRANT command. The two exceptions are CASCADE CONSTRAINTS and FORCE; the GRANT command doesn't have these two options.

- **CASCADE CONSTRAINTS.** This option causes any referential integrity constraints that have been defined using the REFERENCES privilege to be dropped as well.
- **FORCE.** This option forces the revocation of EXECUTE privileges on user-defined object types, where the revokee has tables that depend on those types.

The following REVOKE command revokes the object privileges granted in the previous section:

```
REVOKE SELECT ON AQUATIC_ANIMAL FROM harold;
REVOKE SELECT ON CHECKUP FROM harold;
REVOKE ALL ON CHECKUP_HISTORY FROM harold;
```

Take note of the use of the `REVOKE ALL` command in the previous example. You can use the `REVOKE ALL` command if you want to save typing or if you can't remember exactly what was granted and you want to revoke everything anyway.

Using Security Manager to manage privileges and roles

Security Manager provides you with an easy-to-use interface for managing privileges and roles. You can use Security Manager to manage both system privileges and roles. You can also use Security Manager to view a list of object privileges that have been granted to various users and roles. However, using Security Manager to grant and revoke object privileges depends on the ability of the object owner to run the application.

Using Security Manager to Grant System Privileges

To grant privileges to or revoke privileges from a user, perform the following steps:

1. Start Security Manager, and log on to the database.
2. Double-click the Users folder to expand the list of users in the database.
3. Click the user of interest. You'll see the definition for that user appear in the right pane.
4. Click the System Privileges tab. Your screen will now look similar to the one shown in Figure 11-6.

The list at the top of the right pane contains all possible system privileges. The list at the bottom shows the privileges assigned to the user in question. The arrows in between the lists allow you to move privileges back and forth. To grant a privilege to a user, click that privilege in the top list, and then click the Down Arrow button. The privilege will be moved to the bottom list. Reverse the process to revoke a privilege.

When you have finished editing a user's privileges, click the Apply button to make your changes permanent. If you make a mistake during the editing process, you can press the Revert button to cancel all the edits that you've made.

You can manage roles by using Security Manager in the same manner as system privileges. Just click the Role tab.

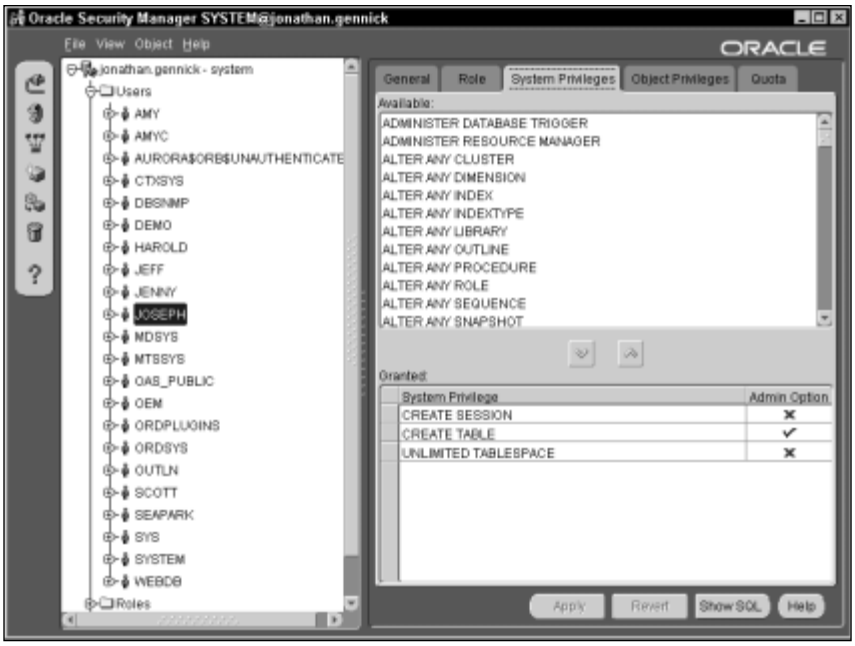

Figure 11-6: Editing a user's system privileges

Using Security Manager to Grant Object Privileges

Although it's possible to grant object privileges using Security Manager, when you try to do so, you'll find yourself in this catch-22 situation: Only an object's owner can grant privileges on an object, but only privileged users, such as DBAs, can run Security Manager.

You probably don't want to grant the DBA role to all your users just so they can use Security Manager to grant object privileges, so what are you to do? One solution to this problem is to grant your users the role named `SELECT_CATALOG_ROLE`. Another is to grant them the `SELECT ANY TABLE` system privilege. Either solution allows them to select from the data dictionary views, which allows them to run Security Manager.

Granting `SELECT_CATALOG_ROLE` is preferable to granting `SELECT ANY TABLE`, but there have been problems with the `SELECT_CATALOG_ROLE` not working for some releases of Oracle Enterprise Manager. The `SELECT ANY TABLE` privilege gives a user access to *all* tables in the database, including those owned by other users. Consequently, you must consider security implications before granting this privilege to a user.

Once a user has access to the data dictionary views, he or she can use Security Manager to grant other users access to his or her tables, views, and so forth. Just follow these steps:

1. Run Security Manager, and log on to the database.
2. Navigate to the user to whom you want to grant privileges, and highlight that user. The right pane will change to allow you to edit information for that user.
3. Click the Object Privileges tab. Your screen should look similar to the one shown in Figure 11-7.
4. Grant whatever object privileges are necessary, and then click the Apply button.

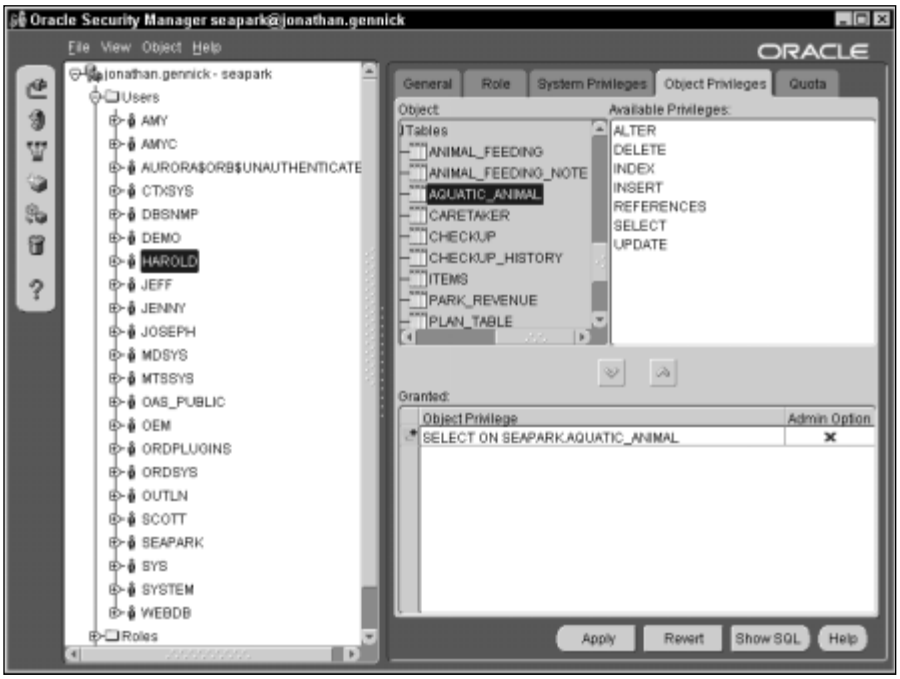

Figure 11-7: Granting the SELECT privilege on the AQUATIC_ANIMAL table to user HAROLD

Other than the need to log on as the object owner, the process you use for managing object privileges is identical to the process you use for managing system privileges.

Creating Roles

You use roles to pull together sets of privileges, such as access to tables, for easier management. Consider the task of creating 1,000 users and granting the following privileges to each:

```
CREATE TABLE
CREATE SEQUENCE
CREATE VIEW
CREATE SYNONYM
CREATE SESSION
ALTER SESSION
CREATE CLUSTER
CREATE DATABASE LINK
```

Granting these privileges individually would be a daunting task. It would be more so if someone asked you to go back later and grant `CREATE TRIGGER` to each of those same users. Your task would be easier if you created a role, say the `CONNECT` role, granted those privileges to the role, and then granted that one role to your users.

When you grant a role to a user, that user inherits all privileges granted to the role. A user can be assigned any number of roles. A role can be assigned any number of privileges. Roles simplify the tasks of managing the privileges granted to users. They allow changes to be made at one central point — the role — rather than having to be made to each user individually.

The initialization parameter `max_enabled_roles` limits the number of enabled roles that may be held by a user at any given time. Make sure that this is set high enough to accommodate the number of roles that you will be granting to your users.

Creating a role

You use the `CREATE ROLE` command to create a role. The syntax looks like this:

```
CREATE ROLE role_name [IDENTIFIED BY password];
```

If you wish to require a password, use the `IDENTIFIED BY` clause. This requires a user to enter a password before using the privileges conveyed by the role. Besides using passwords, there are some other role authentication options that aren't described here. See Appendix A, "SQL Statement Reference," for details.

You must log on as a user with DBA authority or the CREATE ROLE privilege to create roles. To modify roles, you must have the ALTER ANY ROLE privilege. To remove a role, you must have the DROP ANY ROLE privilege. Role names are subject to the naming rules for Oracle objects. Refer to the sidebar "Choosing Names for Users and Passwords" for a quick summary of object-naming guidelines.

As an example, to create a role for the SEAPARK application administrator, you might issue this command:

```
CREATE ROLE seapark_administrator;
```

Of course, a role isn't very useful until you've had a chance to grant some privileges to that role. You'll see how to do that in the next section.

Adding privileges to a role

The task of granting privileges to a role is no different than the task of granting privileges to a user. Use the GRANT command; the syntax is the same. For example, the following commands grant some system privileges to the SEAPARK_ADMINISTRATOR role:

```
GRANT CREATE USER, GRANT ANY ROLE TO seapark_administrator;
GRANT CREATE SESSION TO seapark_administrator WITH ADMIN OPTION;
```

You can grant two types of privileges to a role. The example just shown grants system privileges. It's also common to grant object privileges to roles. Object privileges allow access to objects such as tables and views, and must be granted by the owner of the objects in question. For example, the application administrator might log on as the SEAPARK user and make the following grants:

```
GRANT SELECT ON caretaker TO seapark_administrator;
GRANT SELECT ON aquatic_animal TO seapark_administrator;
```

After you've created a role and granted privileges to that role, you are ready to grant that role to the users who need it. You've already seen how to use the GRANT command to do that earlier in this chapter.

Removing (revoking) privileges from roles

Use the REVOKE command to revoke a privilege from a role. To revoke system privileges or roles from a role, the syntax looks like this:

```
REVOKE {system privilege|role} FROM role;
```

Object privileges, such as SELECT, INSERT, and so forth, must be removed by the object's owner. The syntax is different because you have to supply the name of the object involved. It looks like this:

```
REVOKE object privilege ON object name FROM role;
```

As an example, the following statements revoke the privileges granted by the examples in the previous section:

```
REVOKE CREATE USER, GRANT ANY USER,
       CREATE SESSION
FROM seapark_administrator;
REVOKE SELECT ON caretaker FROM seapark_administrator;
REVOKE SELECT ON caretaker FROM seapark_administrator;
```

Using Security Manager to create roles

You can create roles and delete them by using Security Manager in much the same manner as you create and delete users. You can create a role in Security Manager by right-clicking the Roles folder and selecting Create from the pop-up menu. This will display the Create Role dialog box shown in Figure 11-8.

Figure 11-8: Using Security Manager to create a role

You can use the Role, System Privileges, and Object Privileges tabs to grant roles, system privileges, and object privileges, respectively, to the new role. When you're done, click the Create button, and Security Manager will execute the necessary SQL statements to create the role as you have defined it.

Creating and Assigning User Profiles

Profiles, like roles, can simplify and streamline your work. A profile is a collection of parameters given a name and assigned to one or more Oracle users. For the most part, profiles place limits on what a user can do. Profiles allow you to limit the system resources used by a particular group of users. For example, you can cause the database to end a query that is executing for more than one hour.

Profiles, once created, can be assigned to users. Oracle has one profile preloaded with its default database. This profile is named `DEFAULT`. Unless you specify otherwise, all new users are assigned the `DEFAULT` profile.

Creating and assigning a new profile

The syntax shown in Listing 11-1 creates a new profile.

Listing 11-1: Syntax for creating a new profile

```
CREATE PROFILE profile_name
[SESSIONS_PER_USER { n | UNLIMITED | DEFAULT } ]
[CPU_PER_SESSION { n | UNLIMITED | DEFAULT } ]
[CPU_PER_CALL { n | UNLIMITED | DEFAULT } ]
[CONNECT_TIME { n | UNLIMITED | DEFAULT } ]
[IDLE_TIME { n | UNLIMITED | DEFAULT } ]
[LOGICAL_READS_PER_SESSION { n | UNLIMITED | DEFAULT } ]
[LOGICAL_READS_PER_CALL { n | UNLIMITED | DEFAULT } ]
[PRIVATE_SGA { n [ K | M ] | UNLIMITED | DEFAULT } ]
[FAILED_LOGIN_ATTEMPTS { n | UNLIMITED | DEFAULT } ]
[PASSWORD_LIFE_TIME { n | UNLIMITED | DEFAULT } ]
[PASSWORD_REUSE_TIME { n | UNLIMITED | DEFAULT } ]
[PASSWORD_REUSE_MAX { n | UNLIMITED | DEFAULT } ]
[PASSWORD_LOCK_TIME { n | UNLIMITED | DEFAULT } ]
[PASSWORD_GRACE_TIME { n | UNLIMITED | DEFAULT } ]
[PASSWORD_VERIFY_FUNCTION  [ function_name | NULL | DEFAULT ] ]
[COMPOSITE_LIMIT [n | UNLIMITED | DEFAULT ] ];
```

When you select `DEFAULT` for any of the parameters, that profile parameter receives the value of the same parameter in the default profile. Notice that you can control password parameters in the syntax for creating a new profile. The upcoming subsection “Using a profile to manage password features” provides more information.

You specify time for the password limits in days. Use fractions to represent periods of less than one day. For example, one hour is expressed as 1/24. CPU times are expressed in hundredths of a second. So a `CPU_PER_SESSION` limit of 100 allows the use of one second of CPU time.

After creating a profile, you can assign it to a user. Use the following syntax to do that:

```
ALTER USER username PROFILE profile_name;
```

Replace *username* in this syntax with the name of a database user. Replace *profile_name* with the name of a profile.

Using a profile to manage password features

Oracle implements several features that give you control over how users manage their passwords. These features include the ability to do the following:

- ✦ Expire a password
- ✦ Prevent reuse of passwords
- ✦ Enforce password complexity

Expiring a User's Password

You need to use profiles to take advantage of these features, with one exception: expiring a user's password. The command to do that is user-specific and looks like this:

```
ALTER USER username PASSWORD EXPIRE;
```

When you expire a user's password, the user will still be able to log on, but upon logon, he or she will be forced to immediately choose a new password.

Preventing Password Reuse

You can prevent users from reusing old passwords by assigning them to a profile that keeps a password history. You can choose to base the history list on either the number of days before reuse is allowed or the number of passwords before reuse is allowed.

To retain a user's password history for a specific number of days, you must set the `PASSWORD_REUSE_TIME` parameter in the user's profile. Note that if you set `PASSWORD_REUSE_TIME` to a specific number of days, you must also set `PASSWORD_REUSE_MAX` to `UNLIMITED`. In addition to altering a profile, you can set these values when you create the profile. The following example creates a profile named `SEAPARK_USERS` that retains a 180-day password history:

```
CREATE PROFILE seapark_users
    LIMIT PASSWORD_REUSE_TIME 180
          PASSWORD_REUSE_MAX UNLIMITED;
```

Rather than have the retention period based on a period of time, you can choose to base it on a fixed number of passwords. The following example changes the SEAPARK_USERS profile so that the history for a user always includes the ten most recently used passwords:

```
ALTER PROFILE seapark_users
    LIMIT PASSWORD_REUSE_MAX 10
          PASSWORD_REUSE_TIME UNLIMITED;
```

These two methods for determining password retention time are mutually exclusive. You may limit reuse based on elapsed days, or on the number of passwords, but you can't use both methods at once in the same profile.

Enforcing Password Complexity

You can enforce password complexity by specifying the name of a function that Oracle calls whenever a user changes a password. The function can check to be sure that the password is really two words, that it contains one or more nonalpha characters, and so forth. You can use a third-party function, or you can write your own.

To specify a password complexity verification function, you need to modify a profile and supply the function name as the value for the PASSWORD_VERIFY_FUNCTION setting. For example, the following command makes the SEAPARK_PW function the complexity verification function for the SEAPARK_USERS profile:

```
ALTER PROFILE seapark_users LIMIT
        PASSWORD_VERIFY_FUNCTION seapark_pw;
```

Password verification functions must be owned by the user named SYS, and the function header must match this format:

```
function_name (
    username IN VARCHAR(30),
    password IN VARCHAR (30),
    old_password IN VARCHAR (30)
)RETURN BOOLEAN
```

The function returns a value of TRUE if the user's new password is acceptable. Otherwise, it should return FALSE. The syntax for the CREATE PROFILE and ALTER PROFILE commands is almost identical. Any clause that you've seen used with ALTER PROFILE may also be used with CREATE PROFILE, and vice-versa.

Oracle does provide a working password template function that can be modified to meet your specific requirements. Using this is a whole lot easier than writing one from scratch, especially if you're not familiar with PL/SQL. The file containing the template function is $ORACLE_HOME/rdbms/admin/utlpwdmg.sql.

Using Security Manager to manage profiles

Instead of using commands, you can use Security Manager to manage profiles. The process to create, edit, and delete profiles is the same as the process used to create, edit, and delete users and roles.

To edit an existing profile using Security Manager, just navigate to that profile, highlight it, and edit its properties in the dialog box that appears in the right pane of the Security Manager window. Figure 11-9 shows Security Manager being used to edit the SEAPARK_USERS role:

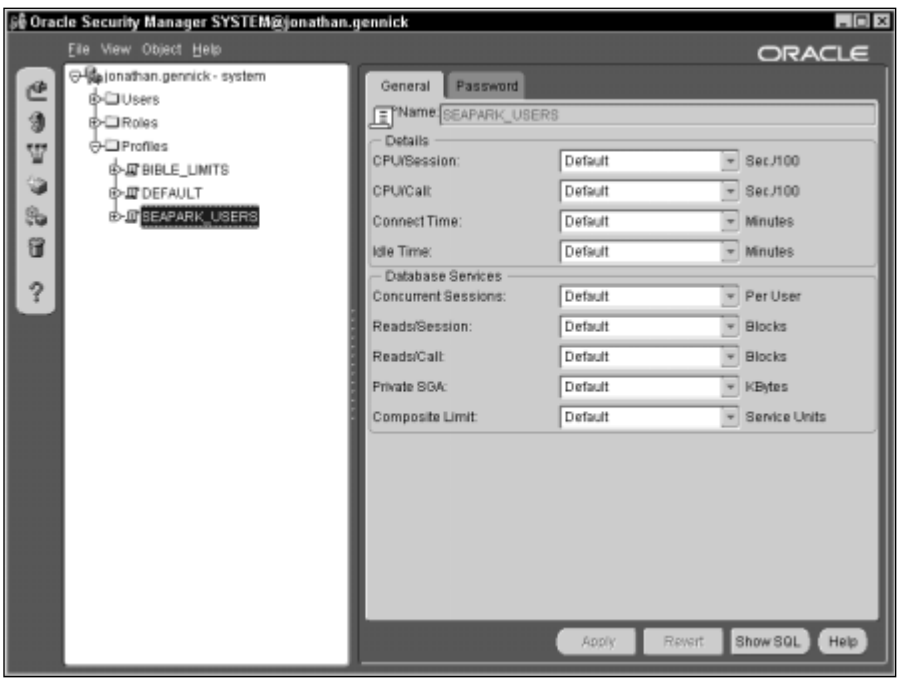

Figure 11-9: Editing the SEAPARK_USERS profile

To create a new profile, right-click the Profiles folder and select Create. The tabs in the Create Profile dialog box are the same as those used to edit a profile. Profiles are assigned to users when you create them. Just choose the appropriate profile from the drop-down list on the General tab of the Create User dialog box.

Becoming a User

When troubleshooting a security problem for a user, it's sometimes helpful to log on as the user involved. However, you may not know the user's password. Asking users for their password is a bad practice because it gets them used to giving it out on request. Before you know it, they'll be giving it out to someone who is not authorized to have it.

Temporarily changing a user's password

So how do you log on as another user when you need to? While Oracle won't let you query the data dictionary to find a user's password in clear text form, you can retrieve the encrypted version of a password. You can then temporarily change the user's password. Later, using the undocumented `IDENTIFIED BY VALUES` clause of the `ALTER USER` command, you can change the password back to its original value, by following these steps:

1. Retrieve the user's encrypted password from the `DBA_USERS` view. The password will be in the form of a string of hexadecimal digits. Write this down or otherwise save it so that you can restore it later.
2. Change the user's password to something you do know. Use the `ALTER USER` command to do this.
3. Log on as the user in question, using the temporary password that you set.
4. Using the hexadecimal password that you saved, issue an `ALTER USER` command to change the password back to its original value. Use the `IDENTIFIED BY VALUES` clause, as shown in this example:

   ```
   ALTER USER joseph IDENTIFIED BY VALUES 'AD923596D59E92AA'
   ```

When you alter a user's password in this manner, the password history is not checked, the expiration date is not reset, and the complexity function is not called. However, Step 2, where you change the user's password to a temporary value, will cause all these to be done. Your temporary password will be added to the user's password history list, and the password expiration date will be set based on when you set the temporary password.

A become_user script

The code in Listing 11-2 represents one possible approach to the task of writing a SQL*Plus script that allows you to become another user.

Listing 11-2: **A SQL*Plus script to become another user**

```
ACCEPT username CHAR PROMPT 'Become User >'
ACCEPT temp_pass CHAR PROMPT 'Temporary Password >'

SET TERMOUT OFF
COLUMN password NOPRINT NEW_VALUE current_password
SELECT password
FROM dba_users
WHERE username=UPPER('&&username');
SET TERMOUT ON

ALTER USER &&username IDENTIFIED BY &&temp_pass;
CONNECT &&username/&&temp_pass

PAUSE Press ENTER to reset the password back to what it was.
ALTER USER &&username
IDENTIFIED BY VALUES '&&current_password';
```

This script uses the `COLUMN` command's `NEW_VALUE` clause to store the user's current password in a SQL*Plus user variable named `current_password`. Only the encrypted hex representation of the password will be stored. The script then changes the user's password, in the normal fashion, to the temporary value that you specified, and logs you on to the database as that user. Listing 11-3 shows this script being used to temporarily log on as the `SEAPARK` user:

Listing 11-3: **Temporarily logging on as SEAPARK**

```
SQL> connect system/manager
Connected.
SQL> @become_user
Become User >seapark
Temporary Password >temp
old   1: ALTER USER &&username IDENTIFIED BY &&temp_pass
new   1: ALTER USER seapark IDENTIFIED BY temp

User altered.

Connected.
Press ENTER to reset the user's password back to what it was.

old   1: ALTER USER &&username IDENTIFIED BY VALUES '&&current_password'
new   1: ALTER USER seapark IDENTIFIED BY VALUES 'E42711E67FF7A82E'

User altered.
```

The PAUSE command in the script allows you to control exactly when the password is reset to its original value. This gives you the ability to also run any third-party software that the user may be using and to connect to the database using the temporary password. Then, after you connect, you press Enter on your SQL*Plus session, and the user's password is reset.

Viewing Users, Security, and the Data Dictionary

If you need to find out information about users, roles, and profiles in your database, you can query Oracle's data dictionary. Several data dictionary views return information about profiles, users, and the privileges that you have granted to those users. These views include:

DBA_USERS	Returns one row for each database user. Use this view to retrieve information about account status, password expiration, default and temporary tablespace settings, and profile name.
DBA_ROLES	Returns one row for each role defined in the database.
DBA_PROFILES	Returns one row for each user profile setting. Use this view when you want to list profiles that have been defined, or when you want to list the settings defined for a particular profile.
DBA_SYS_PRIVS	Returns one row for each system privilege that has been granted to a user or to a role.
DBA_TAB_PRIVS	Returns one row for each object privilege held by a user. Use this parameter when you want to see a list of objects to which a user has access.
DBA_ROLE_PRIVS	Returns one row for each role granted to a user or for each role granted to another role.

Listing users in the database

The DBA_USERS view provides information about the users that exist in your database. You can use this view to generate a list of users by executing a query such as this:

```
SELECT username, account_status, profile
FROM dba_users
ORDER BY username;
```

You can also use this view to find out the default and temporary tablespace settings for a user.

Listing privileges granted to a user

When it comes to discovering the privileges held by a user, there are four areas that you need to look at:

- ✦ System privileges granted directly to the user
- ✦ Roles granted directly to the user
- ✦ System privileges and roles granted to roles held directly by the user
- ✦ Object privileges granted to the user

The third item in the list can be the most difficult to track down. When you discover that a user has been granted a role, to get a complete picture of the user's privileges, you need to in turn find out what has been granted to that role. In the process of doing that, you might discover yet another role to look at, and so forth.

Listing System Privileges Held by a User

To list the system privileges held directly by a user, query the DBA_SYS_PRIVS view. For example, the query in the following example returns all of the privileges granted directly to the user named SYSTEM:

```
SQL> SELECT privilege, admin_option
  2  FROM dba_sys_privs
  3  WHERE grantee='SYSTEM';

PRIVILEGE                                ADM
---------------------------------------- ---
UNLIMITED TABLESPACE                     YES
```

In this case, the SYSTEM user holds the UNLIMITED TABLESPACE privilege with the ADMIN option. The ADMIN option allows the SYSTEM user to grant this privilege to other users. This may not seem like much considering that the user in question is SYSTEM, and it's not. The SYSTEM user gets most of its privileges via the DBA role. To find out what privileges the DBA role conveys, you would also query the DBA_SYS_PRIVS view, as shown in Listing 11-4.

Listing 11-4: Querying for DBA privileges

```
SQL> SELECT privilege, admin_option
  2  FROM dba_sys_privs
  3* WHERE grantee='DBA';

PRIVILEGE                                ADM
---------------------------------------- ---
ADMINISTER DATABASE TRIGGER              YES
ADMINISTER RESOURCE MANAGER              YES
```

```
ALTER ANY CLUSTER                       YES
ALTER ANY DIMENSION                     YES
ALTER ANY INDEX                         YES
ALTER ANY INDEXTYPE                     YES
...
...
...
```

This is a much longer list. It still doesn't give you the complete picture, though, because the DBA role also conveys some roles that have been granted to it.

Listing Roles Held by a User

Roles may be held by a user or by another role. You can list these roles by querying the DBA_ROLE_PRIVS view. The following example shows how to query for roles held by the DBA role:

```
SQL> SELECT granted_role, admin_option
  2  FROM dba_role_privs
  3  WHERE grantee='DBA';

GRANTED_ROLE                   ADM
------------------------------ ---
DELETE_CATALOG_ROLE            YES
EXECUTE_CATALOG_ROLE           YES
EXP_FULL_DATABASE              NO
IMP_FULL_DATABASE              NO
SELECT_CATALOG_ROLE            YES
```

The ADMIN option, represented here by the column titled ADM, indicates whether the role can be further granted to other users. Use the same approach for listing the roles held directly by a user: Simply compare the grantee column to a username instead of a role name.

Listing Object Privileges Held by a User

The DBA_TAB_PRIVS view returns information about object privileges held by a user. Listing 11-5 demonstrates how you can query this view to find the objects to which a user has been granted direct access.

Listing 11-5: **Querying the DBA_TAB_PRIVS view**

```
SQL> COLUMN object_name FORMAT A30
SQL> COLUMN privilege FORMAT A11
SQL> COLUMN grantable FORMAT A10
```

Continued

Listing 11-5 *(continued)*

```
SQL> SELECT owner || '.' || table_name object_name,
  2         privilege, grantable
  3  FROM dba_tab_privs
  4  WHERE grantee = 'SYSTEM'
  5  ORDER BY owner, table_name, privilege;

OBJECT_NAME   PRIVILEGE       GRANTABLE
------------------------------ ----------- ----------
SYS.DBMS_AQ                    EXECUTE     YES
SYS.DBMS_AQADM                 EXECUTE     YES
SYS.DBMS_AQ_IMPORT_INTERNAL    EXECUTE     YES
SYS.DBMS_DEFER_IMPORT_INTERNAL EXECUTE     NO
SYS.DBMS_RULE_EXIMP            EXECUTE     YES
```

The COLUMN commands in this example are not strictly necessary, but the nature of this particular query makes for some very long lines of output. The COLUMN commands limit the display width of the columns so that the output isn't wrapped around on the screen. It's a lot easier to read this way.

Object privileges may be granted to roles as well as to users. In this example, the user of interest was SYSTEM, but you can substitute any other username or role name in its place.

Listing information about roles

You can generate a list of roles defined in the database by querying the DBA_ROLES view. For example:

```
SQL> SELECT role
  2  FROM dba_roles;

ROLE
------------------------------
CONNECT
RESOURCE
DBA
SELECT_CATALOG_ROLE
```

To find out which privileges have been granted to a role, follow the same process described earlier for listing privileges granted to a user.

Listing the definition for a profile

To see the definition of a user profile, you can query the DBA_PROFILES view. The DBA_PROFILES view contains the following columns:

```
PROFILE             NOT NULL VARCHAR2(30)
RESOURCE_NAME       NOT NULL VARCHAR2(32)
RESOURCE_TYPE                VARCHAR2(8)
LIMIT                        VARCHAR2(40)
```

The columns contain the following data:

PROFILE	Contains the name of the profile
RESOURCE_NAME	Contains the name of a resource limit that you can specify as part of the CREATE PROFILE command
RESOURCE_TYPE	Will be either KERNEL or PASSWORD, depending on whether the resource has to do with password management
LIMIT	Contains the resource limit as specified in the CREATE PROFILE command

For each profile that you define, the DBA_PROFILES view will return several rows, one for each resource limit in the profile. Listing 11-6 demonstrates how you can return the definition of the profile named DEFAULT.

Listing 11-6: **Retrieving the definition for a profile**

```
SQL> SELECT resource_name, limit
  2  FROM dba_profiles
  3  WHERE profile='DEFAULT';

RESOURCE_NAME                    LIMIT
-------------------------------- ----------------
COMPOSITE_LIMIT                  UNLIMITED
SESSIONS_PER_USER                UNLIMITED
CPU_PER_SESSION                  UNLIMITED
CPU_PER_CALL                     UNLIMITED
LOGICAL_READS_PER_SESSION        UNLIMITED
LOGICAL_READS_PER_CALL           UNLIMITED
IDLE_TIME                        UNLIMITED
CONNECT_TIME                     UNLIMITED
PRIVATE_SGA                      UNLIMITED
FAILED_LOGIN_ATTEMPTS            UNLIMITED
PASSWORD_LIFE_TIME               UNLIMITED
PASSWORD_REUSE_TIME              UNLIMITED
```

Continued

Listing 11-6 *(continued)*

```
PASSWORD_REUSE_MAX              UNLIMITED
PASSWORD_VERIFY_FUNCTION        UNLIMITED
PASSWORD_LOCK_TIME              UNLIMITED
PASSWORD_GRACE_TIME             UNLIMITED

16 rows selected.
```

The DBA_PROFILES view will return a row for each possible resource limit, regardless of whether you specified all of them when you created the profile.

Summary

In this chapter, you learned:

- ✦ You use system privileges to control what a user can and can't do to the database. Object privileges control what a user can and can't do to individual objects such as a table or a view.
- ✦ Roles enable the DBA and the application developer to simplify privilege management by lumping related privileges together under a single role. Subsequently, a new user can be assigned (granted) to a single role rather than assigned numerous privileges individually.
- ✦ You can use profiles to limit a user's consumption of CPU resources and to enforce rules regarding password complexity. Profiles also define how often passwords need to be changed.
- ✦ When you create a user, always define both a default and a temporary tablespace. Don't forget to provide quota on the default tablespace.

✦ ✦ ✦