Chapter 13

SECURITY

- ❑ Users and privileges
- ❑ Object privileges
- ❑ Granting object privileges
- ❑ Using synonyms
- ❑ Revoking object privileges
- ❑ System privileges
- ❑ Using roles
- ❑ ALTER USER command
- ❑ Data dictionary views

Users and Privileges

Every user in Oracle must have a valid username and password. In order to access Oracle database, one must logon by using username and password.

At the time of creation of database, Oracle creates two users – SYSTEM and SYS. These two users have privileges to perform administrative operations such as creating users, altering and dropping users, roles, profile, tablespaces (pieces that make up the database) etc.

However, it is possible to create new users using CREATE USER command as follows:

```
SQL> create user srikanth identified by oraclebook;

User created.
```

The above command creates a new user with the name SRIKANTH and password ORACLEBOOK.

Note: *In order to create a new user you must logon either as SYSTEM or as SYS.*

Though the account with the name SRIKANTH is created, it cannot even logon now as shown below.

```
SQL> connect srikanth/oraclebook;
ERROR:
ORA-01045: user SRIKANTH lacks CREATE SESSION privilege; logon denied
```

CONNECT command is used to move to another user from current user. In the above example, we were moving from SYSTEM to SRIKANTH.

The error indicates that user SRIKANTH lacks CREATE SESSION privilege. CREATE SESSION is a system privilege, which allows user to create a session with Oracle (logon).

Note: *A session with Oracle start from the point of login and ends when user logs out.*

In order to permit user to create a session and perform other operations such as creating tables, view etc., user must be granted CONNECT and RESOURCE roles as follows.

```
SQL> grant connect, resource to srikanth;

Grant succeeded.
```

We will discuss more about roles later in this chapter. For the time being it is sufficient to know that these two roles will enable SRIKANTH to connect to Oracle database and also allow him to create and use object.

Privilege

A privilege is a right to access an object such as a table, view etc., or to execute a particular type of SQL command such as CREATE TABLE.

Privileges are classified into two categories depending upon what type of right they confer with the user.

- System privileges
- Object Privileges

System privilege

A system privilege is a right to perform certain operation in the system. For example, CREATE SESSION privilege allows user to create a session with Oracle, CREATE TABLE privilege allows user to create table and so on.

Generally system privileges are granted to users through roles. Only DBAs are concerned with system privileges.

Object privilege

An object privilege is a right to perform a particular operation on an object. An object either is a table, view, sequence, procedure, function, or package.

The next section will discuss more about object privileges.

Object Privileges

User owns the object that he/she creates. Owner has complete access to the object. For example, if the object is a table then owner can select, insert, delete, update, alter, create an index on table and even drop the table. Whereas other users do not have any access to the object, unless they are granted a privilege explicitly.

The following is the list of object privileges available in Oracle.

Privilege	**What is permitted?**
ALTER	Changing the definition of the object.
DELETE	Deleting rows from the object
EXECUTE	Execute the object.
INDEX	Creating an index on the object.
INSERT	Inserting rows into object
REFERENCES	Referencing the object in foreign key constraint.
SELECT	Selecting rows from the object.
UPDATE	Updating the data of the object.

Table 1: Object Privileges.

As you see in table 1, each object privilege specifies what can be done with an object. But not all object privileges are applicable to all objects. For instance, ALTER privilege is not applicable to views, similarly EXECUTE privilege is applicable only to procedure and functions. For the list of object privileges available on different types of objects, see table 2.

Object	**Privileges Available**
TABLE	SELECT, INSERT, DELETE, UPDATE, ALTER, INDEX, REFERENCES.
VIEW	SELECT, INSERT, UPDATE, AND DELETE.
SEQUENCE	SELECT, ALTER.
PROCEDURE, FUNCTION, PACKAGE and OBJECT TYPE	EXECUTE.

Table 2: Availability of object privileges.

In the next section we will see how to grant privileges to other users so that they can access and perform required operation.

Granting Object Privileges

In order to grant object privileges use GRANT Command.

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

Object_priv is any of the object privileges listed in table 1.

ALL is used to grant all available object privileges on the object.

PUBLIC is used to grant the privilege to all the users of the system.

Now assume user SRIKANTH is the owner of COURSES table and he wants to grant SELECT privilege on COURSES table to user PRANEETH. The following command will do just that.

```
grant select on courses
   to praneeth;
```

The following command grants all available privileges on COMPBATCHES view to PRANEETH.

```
grant all on compbatches
to praneeth;
```

It is possible to restrict the privilege to a few columns in case of UPDATE, INSERT and REFERENCES privileges.

The following command will grant UPDATE privilege to PRANEETH on DURATION column of COURSES table.

```
grant update(duration)
on courses
to praneeth;
```

WITH GRANT OPTION allows the grantee to grant the privilege to other users.

Accessing other user's objects

When a user wants to access a table of other user, the table name is to be preceded by the name of the user who owns the table. Otherwise Oracle assumes that the current user owns the table and if table is not found under current user's account then Oracle displays an error.

For example, if PRANEETH is trying to access COURSES table to which he has been granted SELECT privilege, from his account, the following happens.

```
SQL> select * from courses;
select * from courses
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Oracle assumes COURSE table belongs to user PRANEETH. To specify that COURSES table is belonging to SRIKANTH and not PRANEETH, we have to precede the table name with username.

```
owner.tablename
```

For example, to access COURSES table from user PRANEETH, give the following command.

```
select *
from srikanth.courses;
```

Using synonyms

To simplify accessing tables owned by other users, create a SYNONYM. A synonym is an alias to a table or a view. By creating a synonym you can avoid giving the owner name while accessing tables of others.

The following CREATE SYNONYM command creates a synonym, which is an alias to COURSES table of SRIKANTH.

Remember synonym is to be created in the account of PRANEETH and not in the account of SRIKANTH.

```
SQL>create synonym COURSES for srikanth.courses;

Synonym created.
```

Once a synonym is created, you can use synonym to refer to the table. So to refer to SRIKANTH.COURSES, user PRANEETH may give:

```
SQL>select * from courses;
```

PUBLIC Synonym

If synonym is to be available to all the users of the system, create a public synonym by including option PUBLIC in CREATE SYNONYM command.

The following sequence of commands will create a public synonym that is accessible to all the users in the system.

```
Grant select on courses to public;
```

The above command grants SELECT privilege on COURSES table to users of the database.

Then create a public synonym on SRIKANTH.COURSES so that any user can access the table using the synonym.

```
create public synonym courses
   for srikanth.courses;
```

***Note**: To create a public synonym, you must have the `CREATE PUBLIC SYNONYM` system privilege.*

Now, it is possible for anyone in the system to access SRIKANTH.COURSES table by using the public synonym. For example, user ANURAG can access COURSES table using public synonym as follows:

```
select  * from course;
```

WITH GRANT OPTION

Using option WITH GRANT OPTION with GRANT command allows grantee to grant the privilege that he/she received to other users.

In the following example, SRIKANTH grants SELECT privilege on COURSE to PRANEETH with option WITH GRANT OPTION.

```
grant select on courses
to praneeth
with grant option;
```

Now user PRANEETH can grant SELECT privilege that he has received from SRIKANTH, to ANURAG as follows:

```
grant  select  on srikanth.courses
to  anurag;
```

Now user ANURAG can access COURSES table of SRIKANTH as follows.

```
select * from  srikanth.courses;
```

Note*: Though ANURAG has got privilege from PRANEETH; he has to give SRIKANTH.COURSES to access COURSES because it is owned by SRIKANTH.*

Revoking Object Privilege

To revoke the privileges that were granted earlier, use REVOKE command.

The following is the syntax of REVOKE command.

Both GRANT and REVOKE commands are DCL commands.

```
REVOKE {object_priv | ALL}
       [,{object_priv | ALL} ] ...
   ON [schema.]object
   FROM {user |role|PUBLIC} [,{user|role|PUBLIC}] ...
   [CASCADE CONSTRAINTS]
```

Object_priv is the privilege that is to be revoked. **ALL** revokes all privileges.

CASCADE CONSTRAINTS drops any referential constraint that was based on REFERENCES privilege granted earlier. For example, user A granted REFERENCES privilege to user B on CCODE column of COURSES table. And user B has referred to CCODE of COURSES in references constraint. If user A revokes REFERENCES privilege than references constraint will be deleted, if CASCADE CONSTARINTS options is used.

The following command will revoke SELECT privilege on COURSES from PRANEETH.

```
SQL> revoke  select  on  courses
  2  from  praneeth;
```

Revoking is cascading

When a privilege is revoked from a user, if that user has previously granted that privilege to other users then all grantees of that privilege will also lose that privilege. For example, if A grants a privilege to B and B grants that to C, then both B and C will lose the privilege when A revokes the privilege from B.

What is a Role?

Role is a collection of privileges. In cases where granting privileges user by user and table by table is lengthy, a role can come to your rescue.

The following are the important characteristics of roles:

- A role is a collection of privileges. The privileges may consist of both system and object privileges.
- A role is dynamic. Users of the role will enjoy whatever new privileges (added after the role has been granted to users) the role has been granted. In other words changes made to privileges of the role will affect all the users of the role.
- To create a role one must have CREATE ROLE system privilege.
- Oracle comes with a few predefined roles such as CONNECT, RESOURCE, and DBA.
- A single user can be granted multiple roles.
- User will use only those roles that are enabled by default. However, user can enable/disable any roles that are granted to him.

The following sections will explain how to create and use role.

Creating and using role

A role is created using CREATE ROLE command whose syntax is as follows:

```
CREATE ROLE rolename
   [identified by password]
```

password is the password of the role. Users must use password of the role at the time of enabling the role. Password may be used for extra security.

The following are the three important steps related to roles. Let us see how to create and use a simple role called MANAGER.

Creating a role using CREATE ROLE command. The following command creates a role with the name MANAGER.

```
create role manager;
```

Granting required privileges to role
GRANT command can be used to grant privileges to role. The following GRANT commands are used to grant privilege to MANAGER role.

```
grant select on courses to manager;

grant select, update, insert, delete on batches to manager;

grant all on students to manager;

grant select on payments;
```

Granting role to users
A role can be granted to user using GRANT command as follows.

```
grant manager to praneeth;
grant manager to anurag;
```

Enabling a role
It is possible for a single user to have been granted more than one role. However, all roles granted to user may not be enabled by default. That means, the role is granted to user but user cannot use the privileges of the role.

At the time of creating user or even afterwards, administrator can specify which roles of the user must be enabled by default. The remaining roles are to be enabled explicitly.

If role MANAGER is granted to user but not enabled then user can enable the role as follows:

```
SQL > set role manager;
```

The privileges the user currently has, depend on the roles that are enabled. The roles that are not currently enabled are called as ***disabled roles*** and roles that are currently enabled are called as ***enabled roles.***

For more details on SET ROLE command, please see on-line help.

Dropping a role
It is possible to drop a role using DROP ROLE command. The following command drops role MANAGER that we created earlier.

```
drop role manager;
```

Using password with role

It is possible to assign a password to a role. The password is assigned at the time of creating the role or after the role is created using ALTER ROLE command.

```
create role manager identified by efficient;
```

Then grant a few privileges to MANAGER role. Though DBA can only create the role, any user can grant privileges to role.

```
grant select on courses to manager;
```

Now grant role to user anurag;

```
grant manager to anurag;
```

Now if user ANURAG wants to access COURSES table through the role, first he has to enable the role as follows.

```
set role manager identified by efficient;
```

As the role is assigned a password, user must supply password at the time of enabling role. If user doesn't supply password while enabling role, the following error occurs.

```
SQL> set role manager;
set role manager
*
ERROR at line 1:
ORA-01979: missing or invalid password for role 'MANAGER'
```

ALTER USER Command

ALTER USER command can be used to modify the characteristics of a user. For example it can be used to modify:

- Password
- Default roles

To change the password of the current user, user may give:

```
alter user praneeth identified by tunu;
```

It is possible to specify the default roles of a user using DEFAULT ROLE option of ALTER USER command as follows:

```
alter user book default role all except manager;
```

Note: *Except changing password, user cannot change any other of his/her attributes.*

Please see on-line help for remaining options in ALTER USER command and their usage.

Data Dictionary Views

Data dictionary views contain information regarding database in a simple form so that user can easily understand. All data dictionary views are based on tables owned by user SYS. Data dictionary tables and views are created at the time of creating database.

The following is the list of data dictionary views that are commonly used.

Data dictionary view	**What it contains?**
ALL_TAB_COLUMNS	Columns of all tables and views accessible to the user.
ALL_OBJECTS	Objects accessible to user.
DICTIONARY	Description of data dictionary view.
USER_CATALOG	Tables, views, synonyms and sequences owned by user.
USER_CLUSTERS	Description of user's own clusters.
USER_COL_PRIVS	Grants on columns for which the user is the owner, grantor or grantee.
USER_ERRORS	Current errors on stored objects owned by the user.
USER_INDEXES	Description of the user's own indexes.
USER_OBJECTS	Objects owned by the user.
USER_SEQUENCES	Description of the user's own sequences.
USER_SYNONYMS	The user's private synonyms.
USER_TABLES	Description of the user's own tables.
USER_TAB_PRIVS	Grants on tables for which the user is the owner, grantor, or grantee.
USER_TRIGGERS	Triggers owned by the user.
USER_TYPES	Object types created by user.
USER_USER	Information about the current user.
USER_VIEWS	Text of views owned by the user.

Table 3: Data dictionary views.

The following is the list of synonyms based on Data dictionary views.

Synonym	Data dictionary view
DICT	DICTIONARY
OBJ	USER_OBJECTS
CAT	USER_CATALOG
TABS	USER_TABLES
COLS	USER_TAB_COLUMNS
SEQ	USER_SEQUENCES.
SYN	USER_SYNONYM.
IND	USER_INDEXES.

Table 4: Synonyms for Data dictionary views.

To list all tables owned by current user, enter:

```
SQL> select * from tabs;
```

To list the all the objects and their types, enter:

```
select object_name, object_type from user_objects;
```

To get the query stored along with view, enter:

```
select view_name, text from user_views;
```

Summary

Security is an important feature of any multi-user database system. Oracle implements security using system privileges and object privileges. System privileges specify which commands a user can execute. Unless otherwise specified an object (table, view etc.) can be accessed only by the owner. But using object privileges a user can allow other users to access his objects.

Roles are used to group privileges. When a role is granted to a user, all the privileges that are granted to role are granted to grantee of the role. It greatly simplifies the job of implementing security because with one role many privileges can be granted.

Data dictionary views may be used to get useful information regarding objects, users, and system.

Exercises

1. __________ command is used to change user password.
2. Which object privilege allows user to create an index on the table ______.
3. ________ Option is used to grant a privilege along with permission to grant the privilege to other users.
4. A Role is ________________________________.
5. _______ Command is used to revoke a system privilege.
6. ___________ data dictionary view may be used to know the table that a user can access.
7. _______ data dictionary view is used to know the list of tables owned by the current user.
8. _____ is the synonym for USER_CATALOG data dictionary.
9. Grant UPDATE privilege on STDATE column of BATCHES table to user PRANEETH with permission to grant the privilege to others.
10. Create a role and assign a few privileges to that role. Assign the role to user ANURAG.
11. Enable the role LEADER.
12. Display the table name, column name of all columns that you can access.