Database Programming

Creating and Revoking Object Privileges

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Explain what a ROLE is and what its advantages are
- Construct a statement to create a ROLE and GRANT privileges to it
- Construct a GRANT .. ON .. TO.. WITH GRANT OPTION statement to assign privileges on objects in your schema to other users and/or PUBLIC
- Construct and execute a statement to REVOKE object privileges from other users and/or from PUBLIC
- Distinguish between privileges and roles
- Explain the purpose of a database link

Purpose

If you share a computer with others, whether at school or at home, you've probably had something you're working on or something you've saved either viewed, changed, or deleted by someone else. Wouldn't it be nice to be able to control the privileges others have to your personal files?

For databases, just as at school or home, data security is very important.

Purpose (cont.)

In this lesson, you will learn how to grant or take away access to database objects as a means to control who can alter, delete, update, insert, index, or reference those database objects.

Roles

A role is a named group of related privileges that can be granted to a user. This method makes it easier to revoke and maintain privileges. A user can have access to several roles, and several users can be assigned the same role. Roles are typically created for a database application.

```
SQL> CREATE ROLE manager;
Role created.
SQL> GRANT create table, create view TO manager;
Grant succeeded.
SQL> GRANT manager TO jennifer_cho;
Grant succeeded.
```

Roles (cont.)

To create and assign a role, first the DBA must create the role. Then the DBA can assign privileges to the role, and the role to users.

```
SQL> CREATE ROLE manager;
Role created.
SQL> GRANT create table, create view TO manager;
Grant succeeded.
SQL> GRANT manager TO jennifer_cho;
Grant succeeded.
```

Roles (cont.)

Use the following syntax to create a role:

```
CREATE ROLE role_name;
```

After the role is created, the DBA can use the GRANT statement to assign the role to users as well as assign privileges to the role.

Roles (cont.)

The example shown creates a manager role and then allows managers to create tables and views. It then grants the role to a user. Now the user can create tables and views. If users have multiple roles granted to them, they receive all of the privileges associated with all of the roles.

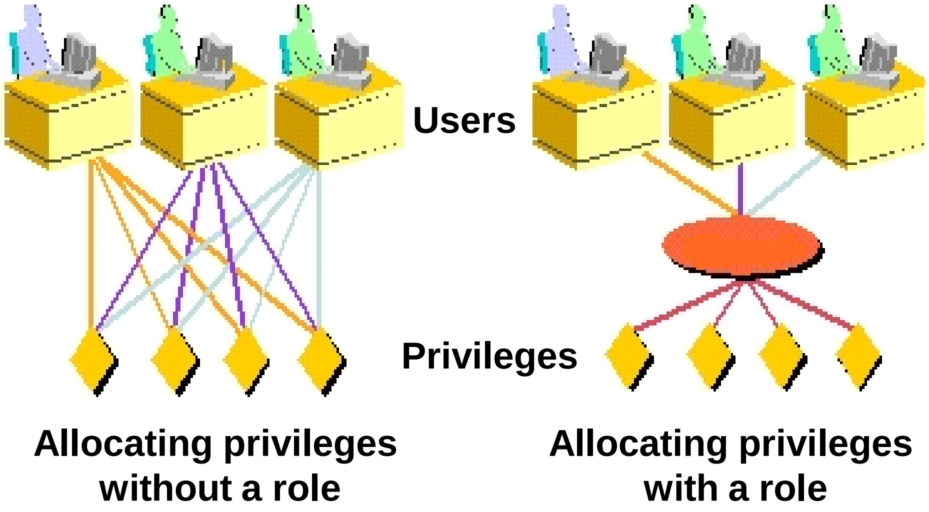
```
SQL> CREATE ROLE manager;
Role created.

SQL> GRANT create table, create view TO manager;
Grant succeeded.

SQL> GRANT manager TO jennifer_cho;
Grant succeeded.
```

Roles (cont.)

Note: The CREATE ROLE is a system privilege that has not been issued to Academy classrooms.

Characteristics Of Roles

- Roles are named groups of related privileges.
- They can be granted to users.
- They simplify the process of granting and revoking privileges.
- They are created by a DBA.


Users
Privileges
Allocating privileges without a role
Allocating privileges with a role

Granting Object Privileges

Use the following syntax to grant object privileges:

```
GRANT object_priv [(column_list)]
ON object_name
TO {user|role|PUBLIC}
[WITH GRANT OPTION];
```

Syntax	Defined
object_priv	is an object privilege to be granted
column_list	specifies a column from a table or view on which privileges are granted
ON object_name	is the object on which the privileges are granted
TO user|role	identifies the user or role to whom the privilege is granted
PUBLIC	grants object privileges to all users
WITH GRANT OPTION	Allows the grantee to grant the object privileges to other users and roles

Object Privileges Guidelines

- To grant privileges on an object, the object must be in your own schema, or you must have been granted the privilege using the WITH GRANT OPTION.
- An object owner can grant any object privilege on the object to any other user or role of the database.
- The owner of an object automatically acquires all object privileges on that object.

GRANT Examples

Scott King (username scott_king) has created a d_songs table. In Example 1 on the right, all users are granted permission to SELECT from Scott's d_songs table. Example 2 grants UPDATE privileges to Jennifer and to the manager role on specific columns in Scott's d_songs table.

```
1. GRANT  select
ON      d_songs
TO    PUBLIC;

2. GRANT  update (title,
artist)
ON      d_songs
TO      jennifer_cho,
manager;

3. SELECT  *
FROM     scott_king.d_song
s;

4. CREATE SYNONYM songs
FOR scott_king.d_songs;

5. SELECT *
FROM songs;
```

GRANT Examples (cont.)

If Jennifer now wants to SELECT data from Scott's table, the syntax she must use is listed in Example 3.

Alternatively, Jennifer could create a synonym for Scott's table and SELECT from the synonym. See the syntax in Examples 4 and 5.

```
1. GRANT  select
ON      d_songs
TO    PUBLIC;

2. GRANT  update (title,
artist)
ON      d_songs
TO      jennifer_cho,
manager;

3. SELECT  *
FROM     scott_king.d_song
s;

4. CREATE SYNONYM songs
FOR scott_king.d_songs;

5. SELECT *
FROM songs;
```

GRANT Examples (cont.)

Different object privileges are available for different types of schema objects. A user automatically has all object privileges for schema objects contained in his schema. A user can grant any object privilege on any schema object that the user owns to any other user or role.

```
1. GRANT  select
ON     d_songs
TO   PUBLIC;

2. GRANT  update (title,
artist)
ON     d_songs
TO     jennifer_cho,
manager;

3. SELECT  *
FROM    scott_king.d_song
s;

4. CREATE SYNONYM songs
FOR scott_king.d_songs;

5. SELECT *
FROM songs;
```

WITH GRANT OPTION

A privilege that is granted using the WITH GRANT OPTION clause can be passed on to other users and roles by the grantee.

Object privileges granted using the WITH GRANT OPTION clause are revoked when the grantor’s privilege is revoked.

WITH GRANT OPTION (cont.)

The example below gives user Scott access to your d_songs table with the privileges to query the table and add rows to the table. The example also allows Scott to give others these privileges:

```
GRANT  select, insert
ON     d_songs
TO     scott_king
WITH   GRANT OPTION;
```

The PUBLIC Keyword

An owner of a table can grant access to all users by using the PUBLIC keyword. The example shown below allows all users on the system to query data from Jason’s d_songs table:

```
GRANT   select
ON      jason_tsang.d_songs
TO      PUBLIC;
```

DELETE Object

If you attempt to perform an unauthorized operation, such as deleting a row from a table on which you do not have the DELETE privilege, the Oracle Server does not permit the operation to take place.

If you receive the Oracle Server error message “table or view does not exist,” you have done one of the following:

- Referenced a table or view that does not exist
- Attempted to perform an operation on a table or view for which you do not have the appropriate privileges

View Privileges

You can access the data dictionary to view the privileges that you have. The table below describes various data dictionary views.

Data Dictionary View	Description
ROLE_SYS_PRIVS	System privileges granted to roles.
ROLE_TAB_PRIVS	Table privileges granted to roles.
USER_ROLE_PRIVS	Roles accessible by the user.
USER_TAB_PRIVS_MADE	Object privileges granted on the user's objects.
USER_TAB_PRIVS_RECD	Object privileges granted to the user.

View Privileges (cont.)

Data Dictionary View	Description
USER_COL_PRIVS_MADE	Object privileges granted on the columns of the user’s objects.
USER_COL_PRIVS_RECD	Object privileges granted to the user on specific columns.
USER_SYS_PRIVS	Lists system privileges granted to the user.

Revoking Object Privileges

You can remove privileges granted to other users by using the REVOKE statement. When you use the REVOKE statement, the privileges that you specify are revoked from the users that you name and from any other users to whom those privileges were granted using the WITH GRANT OPTION clause.

Revoking Object Privileges (cont.)

Use the following syntax to revoke object privileges:

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

CASCADE CONSTRAINTS is required to remove any referential integrity constraints made to the object by means of the REFERENCES privilege.

With Grant Option

The example below revokes SELECT and INSERT privileges given to user Scott on the d_songs table.

```
REVOKE select, insert
ON d_songs
FROM scott_king;
```

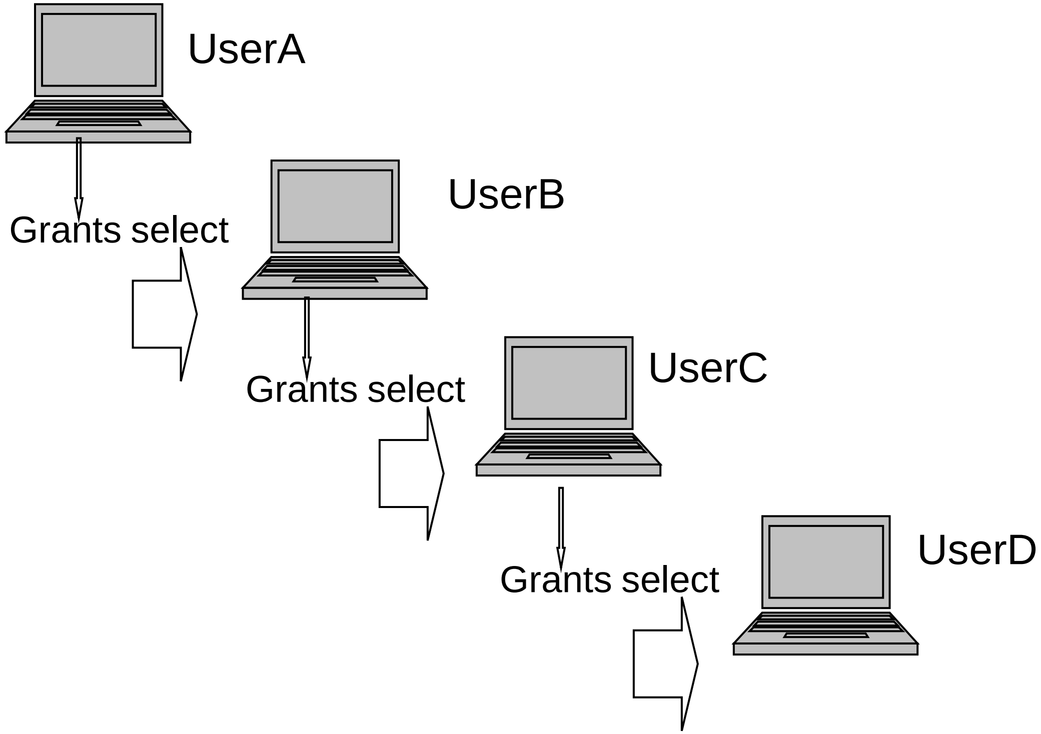
If a user is granted a privilege with the WITH GRANT OPTION clause, that user can also grant the privilege using the WITH GRANT OPTION clause. This means that a long chain of grantees is possible, but no circular grants are permitted.

With Grant Option (cont.)

If the owner revokes a privilege from a user who granted privileges to other users, the revoke statement cascades to all privileges granted.

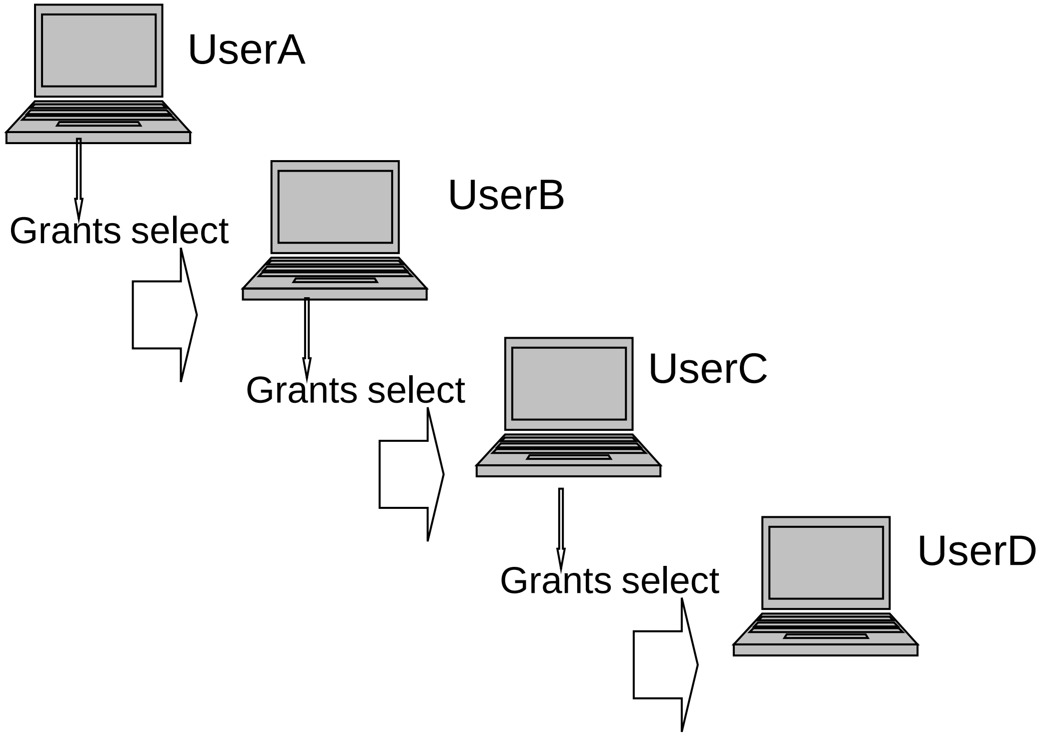
With Grant Option (cont.)

For example, if user A grants SELECT privileges on a table to user B, including the WITH GRANT OPTION clause, user B can grant to user C the SELECT privilege including the WITH GRANT OPTION clause as well.



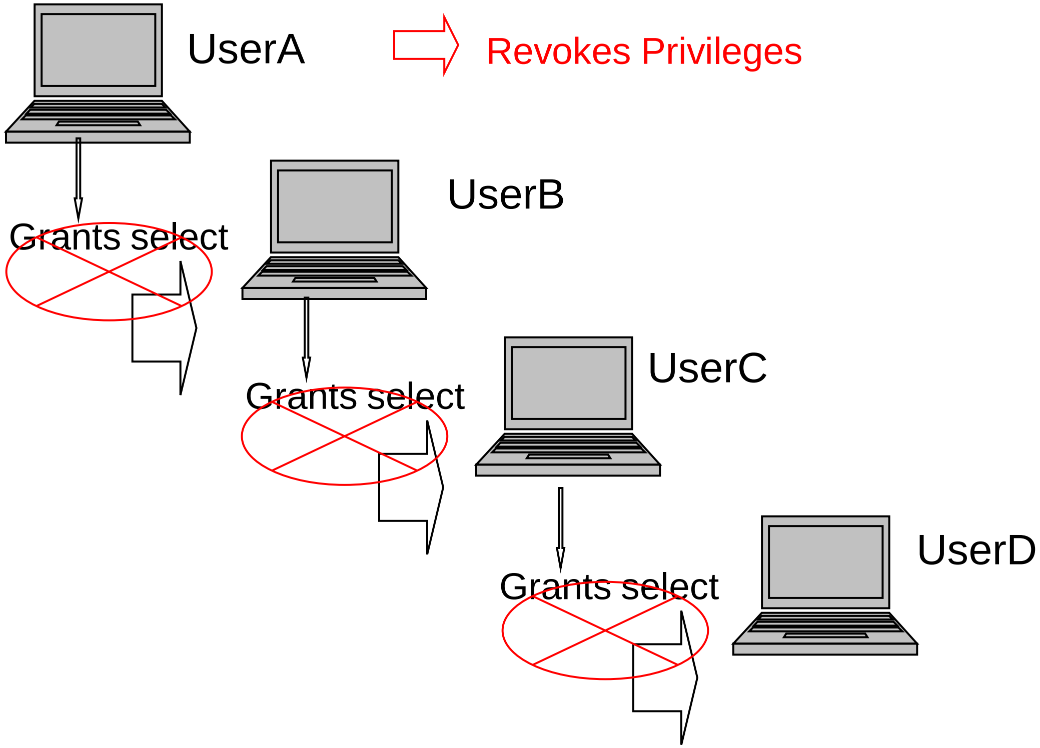
With Grant Option (cont.)

Now, user C can grant to user D the SELECT privilege.



With Grant Option (cont.)

However, if user A revokes privileges from user B, then those privileges granted to users C and D are also revoked.



Private and Public Synonyms

As mentioned earlier in this lesson, you can create a synonym to eliminate the need to qualify the object name with the schema and provide you with an alternative name for a table, view, sequence, procedure, or other object.

Synonyms can be either private (the default) or public. A public synonym can be created by Database Administrators, or database users who have been given the privileges to do so, but not everyone can automatically create public synonyms.

Private and Public Synonyms (cont.)

Note: The CREATE PUBLIC SYNONYM privilege has not been granted to Academy students.

Roles and Privileges

Roles and Privileges differ in a number of ways:

- A user **privilege** is a right to execute a particular type of SQL statement, or a right to access another user's object. All privileges are defined by Oracle.
- **Roles**, on the other hand, are created by users (usually administrators) and are used to group together privileges or other roles. They are created to make it easier to manage the granting of multiple privileges or roles to users.

Roles and Privileges (cont.)

Privileges come with the database and Roles are made by Database Administrators or users of a particular database.

Database Links

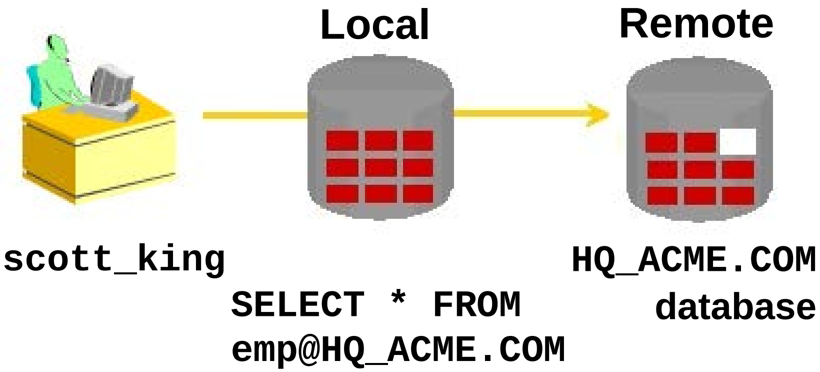
A database link is a pointer that defines a one-way communication path from one Oracle database to another Oracle database.

The link pointer is actually defined as an entry in a data dictionary table. To access the link, you must be connected to the local database that contains the data dictionary entry.

Database Links (cont.)

A database link connection is “one-way” in the sense that a client connected to local database A can use a link stored in database A to access information in remote database B, but users connected to database B cannot use the same link to access data in database A.

CREATE DATABASE LINK – In Oracle Application Express, there is no constant connection to the database, and as a result, this feature is not available.



Database Links (cont.)

If local users on database B want to access data on database A, they must define a link that is stored in the data dictionary of database B.

A database link connection gives local users access to data on a remote database.

For this connection to occur, each database in the distributed system must have a unique global database name. The global database name uniquely identifies a database server in a distributed system.

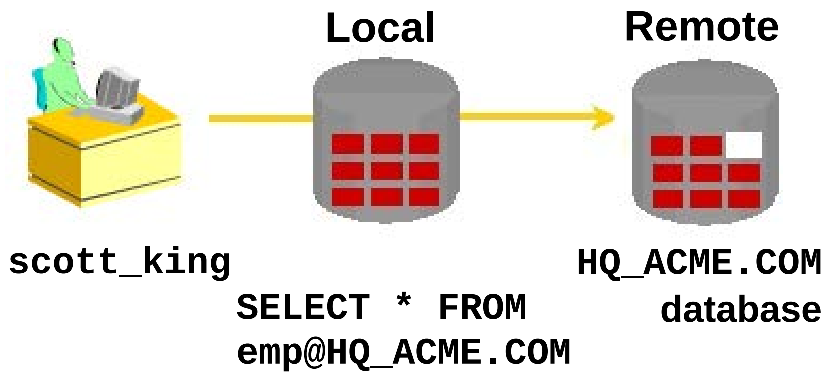
Database Links (cont.)

The great advantage of database links is that they allow users to access another user’s objects in a remote database so that they are bounded by the privilege set of the object’s owner.

In other words, a local user can access a remote database without having to be a user on the remote database.

Database Links (cont.)

The example shows a user scott_king accessing the EMP table on the remote database with the global name HQ.ACME.COM.



Local

Remote

scott_king

```
SELECT * FROM
emp@HQ_ACME.COM
```

HQ_ACME.COM
database

Database Links (cont.)

Typically, the Database Administrator is responsible for creating the database link. The dictionary view USER_DB_LINKS contains information on links to which a user has access.

Once the database link is created, you can write SQL statements against the data in the remote site. If a synonym is set up, you can write SQL statements using the synonym.

Database Links (cont.)

For example:

```
CREATE PUBLIC SYNONYM HQ_EMP
 FOR emp@HQ.ACME.COM;
```

Then write a SQL statement that uses the synonym:
SELECT * FROM HQ_EMP;

You cannot grant privileges on remote objects.

Summary

In this lesson, you should have learned how to:

- Explain what a ROLE is and what its advantages are
- Construct a statement to create a ROLE and GRANT privileges to it
- Construct a GRANT .. ON .. TO.. WITH GRANT OPTION statement to assign privileges to objects in their schema to other users and/or PUBLIC
- Construct and execute a statement to REVOKE object privileges from other users and/or from PUBLIC
- Distinguish between privileges and roles
- Explain the purpose of a database link