module14-2-privileges.htm; updated June 18, 2012; Some figures shown in these notes are from Oracle document D11321GC11.

Module 14-2 – Privileges

Objectives

- Learn the different system and object privileges.
- Learn to grant and revoke privileges.

General

Authentication means to authenticate a system user account ID for access to an Oracle database.

Authorization means to verify that a system user account ID has been granted the right, called a **privilege**, to execute a particular type of SQL statement or to access objects belonging to another system user account.

In order to manage system user access and use of various system objects, such as tables, indexes, and clusters, Oracle provides the capability to grant and revoke **privileges** to individual user accounts.

Example privileges include the right to:

- Connect to a database

- Create a table
- Select rows from another user's table
- Execute another user's stored procedure

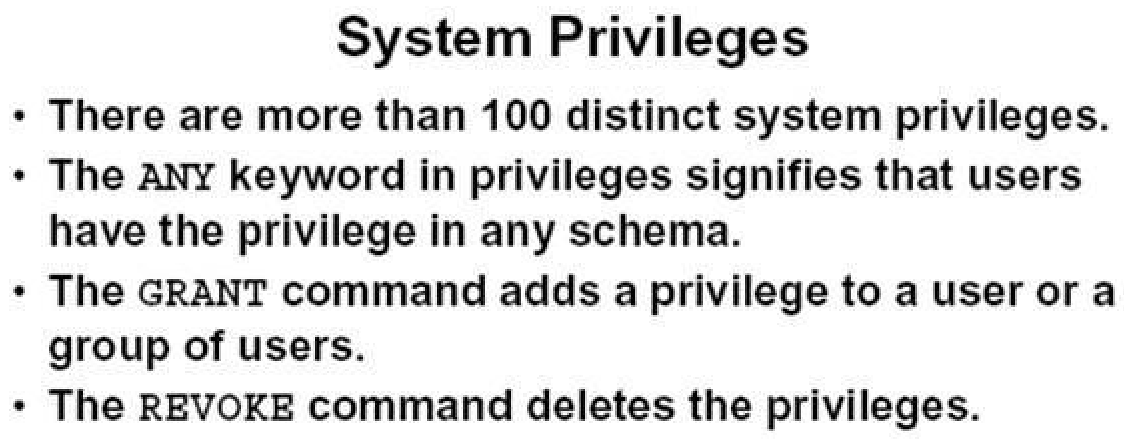
Excessive granting of privileges can lead to situations where security is compromised.

There are **six categories** of privileges:

- **System** privileges allow a system user to perform a specific type of operation or set of operations. Typical operations are creating objects, dropping objects, and altering objects.
- **Schema Object** privileges allow a system user to perform a specific type of operation on a specific schema object. Typical objects include tables, views, procedures, functions, sequences, etc.
- **Table** privileges are schema object privileges specifically applicable to Data Manipulation Language (DML) operations and Data Definition Language (DDL) operations for tables.
- **View** privileges apply to the use of view objects that reference base tables and other views.
- **Procedure** privileges apply to procedures, functions, and packages.
- **Type** privileges apply to the creation of named types such as object types, VARRAYs, and nested tables.

System Privileges

As Oracle has matured as a product, the number of **system** privileges has grown. The current number is over 100. A complete listing is available by querying the view named **SYSTEM_PRIVILEGE_MAP**.


System Privileges

- There are more than 100 distinct system privileges.
- The `ANY` keyword in privileges signifies that users have the privilege in any schema.
- The `GRANT` command adds a privilege to a user or a group of users.
- The `REVOKE` command deletes the privileges.

Privileges can be divided into **three categories**:

- Those enabling system wide operations, for example, CREATE SESSION, CREATE TABLESPACE.
- Those enabling the management of an object that is owned by the system user, for example, CREATE TABLE.
- Those enabling the management of an object that is owned by any system user, for example, CREATE ANY TABLE.

If you can create an object, such as that privilege provided by the **CREATE TABLE** privilege, then you can also drop the objects you create.

Some examples of system privileges include:

Category	**Privilege**
SESSION	Create Session Alter Session
TABLESPACE	Create Tablespace Alter Tablespace Drop Tablespace Unlimited Tablespace
TABLE	Create Table Create Any Table Alter Any Table Drop Any Table Select Any Table
INDEX	Create Any Index Alter Any Index

Some privileges that you might expect to exist, such as **CREATE INDEX**, do not exist since if you can **CREATE TABLE**, you can also create the indexes that go with it and use the **ANALYZE** command.

Some privileges, such as **UNLIMITED TABLESPACE** cannot be granted to a role (roles are covered in Module 14-3)

Granting System Privileges

The command to grant a system privilege is the **GRANT** command. Some example **GRANT** commands are shown here.

```
GRANT ALTER TABLESPACE, DROP TABLESPACE
    TO USER349;


Grant succeeded.


GRANT CREATE SESSION TO USER350
    WITH ADMIN OPTION;

Grant succeeded.
```

In general, you can grant a privilege to either a user or to a role. You can also grant a privilege to **PUBLIC** - this makes the privilege available to every system user.

The **WITH ADMIN OPTION** clause enables the grantee (person receiving the privilege) to grant the privilege or role to other system users or roles; however, you cannot use this clause unless you have, yourself, been granted the privilege with this clause.

The **GRANT ANY PRIVILEGE** system privilege also enables a system user to grant or revoke privileges.

The **GRANT ANY ROLE** system privilege is a dangerous one that you don't give to the average system user since then the user could grant any role to any other system user.

SYSDBA and SYSOPER Privileges

SYSDBA and **SYSOPER** are special privileges that should only be granted to a **DBA**.

This table lists example privileges associated with each of these special privileges.

SYSOPER	**SYSDBA**
STARTUP SHUTDOWN ALTER DATABASE OPEN | MOUNT RECOVER DATABASE ALTER DATABASE ARCHIVELOG RESTRICTED SESSION	SYSOPER PRIVILEGES THAT INCLUDE THE WITH ADMIN OPTION. CREATE DATABASE

ALTER DATABASE BEGIN/END BACKUP	RECOVER DATABASE UNTIL

When you allow database access through a **Password File** using the **REMOTE_LOGIN_PASSWORDFILE** parameter that was discussed in an earlier module, you can add users to this password file by granting them **SYSOPER** or **SYSDBA** system privileges.

You **cannot** grant the **SYSDBA** or **SYSOPER** privileges by using the **WITH ADMIN OPTION**. Also, you must have these privileges in order to grant/revoke them from another system user.

Displaying System Privileges

You can display system privileges by querying the **DBA_SYS_PRIVS** view. Here is the result of a query of the SIUE Oracle database.

```
SELECT * FROM dba_sys_privs

WHERE Grantee = 'USER349';
```

```
GRANTEE                        PRIVILEGE
ADM

------------------------------
---------------------------------------- ---
```

```
USER349                        DROP TABLESPACE
NO

USER349                        ALTER TABLESPACE
NO
```

You can view the users who have **SYSOPER** and **SYSDBA** privileges by querying **v$pwfile_users**. Note: Your student databases will display no rows selected—this output comes from the DBORCL database.

```
SELECT * FROM v$pwfile_users;
```

```
USERNAME        SYSDB SYSOP

--------------- ----- -----

INTERNAL        TRUE  TRUE

SYS             TRUE  TRUE

DBOCK           TRUE  FALSE

JAGREEN         TRUE  TRUE
```

The view **SESSION_PRIVS** gives the privileges held by a user for the current logon session.

Revoking System Privileges

The **REVOKE** command can be used to revoke privileges from a system user or from a role.

Only privileges granted directly with a **GRANT** command can be revoked.

There are no cascading effects when a system privilege is revoked. For example, the DBA grants the **SELECT ANY TABLE WITH ADMIN OPTION** to system **user1**, and then system user1 grants the **SELECT ANY TABLE** to system **user2**, then if system **user1** has the privilege **revoked**, system **user2** still has the privilege.

System Privilege Restrictions

Oracle provides for data dictionary protection by enabling the restriction of access to dictionary objects to the SYSDBA and SYSOPER roles.

For example, if this protection is in place, the **SELECT ANY TABLE** privilege to allow a user to access views and tables in other schemas would not enable the system user to access dictionary objects.

The appropriate **init.ora** parameter is **O7_DICTIONARY_ACCESSIBILITY** and it is set to **FALSE**, SYSTEM privileges allowing access to objects in other schemas would not allow access to the dictionary schema. If it is set **=TRUE**, then access to the **SYS** schema is allowed (this is the behavior of Oracle 7).

Schema Object Privileges

Schema object privileges authorize the system user to perform an operation on the object, such as selecting or deleting rows in a table.

A user account automatically has all object privileges for schema objects created within his/her schema. Any privilege owned by a user account can be granted to another user account or to a role.

The following table provided by Oracle Corporation gives a map of **object** privileges and the type of object to which a privilege applies.

OBJECT PRIVILEGE	**Table**	**View**	**Sequence**	**Procedure**
ALTER	XXX	XXX	XXX	XXX
DELETE	XXX	XXX		
EXECUTE				XXX
INDEX	XXX	XXX		
INSERT	XXX	XXX		
REFERENCES	XXX			
SELECT	XXX	XXX	XXX	
UPDATE	XXX	XXX		

To grant an object privilege, you must specify the privilege and the object. Example commands are shown here.

```
GRANT SELECT, ALTER ON User350.Orders TO PUBLIC;

GRANT SELECT, DELETE ON User350.Order_details TO
user349;
GRANT SELECT ON User350.Order_details
    TO User349 WITH GRANT OPTION;
GRANT ALL ON User350.Order_details
    TO Accountant__Role;
GRANT UPDATE (Price, Description) ON
USER350.Order_details TO User349;
```

Here the **SELECT** and **ALTER** privileges were granted for the **Orders** table belonging to the system user **User350.** These two privileges were granted to **all** system users through the **PUBLIC** specification.

In the 3rd example, **User349** receives the **SELECT** privilege on **User350's Order_Details** table and can also grant that privilege to other system users via the **WITH GRANT OPTION**.

In the 4th example, the **Accountant_Role** role receives **ALL** privileges associated with the **Order_Details** table.

In the 5th example **UPDATE** privilege is allocated for only two columns (**Price** and **Description**) of the **Order_Details** table.

Notice the difference between **WITH ADMIN OPTION** and WITH **GRANT OPTION** - the first applying to System privileges (these are administrative in nature), the second applying to Object privileges.

Revoking Schema Object Privileges

Object privileges are revoked the same way that system privileges are revoked.

Several example REVOKE commands are shown here. Note the use of ALL (to revoke all object privileges granted to a system user) and ON (to identify the object).

```
REVOKE SELECT ON dbock.orders FROM User350;

REVOKE ALL on User350.Order_Details FROM User349;
REVOKE ALL on User350.Order_Details FROM User349
    CASCADE CONSTRAINTS;
```

In the latter example, the **CASCADE CONSTRAINTS** clause would drop referential integrity constraints defined by the revocation of **ALL** privileges.

There is a difference in how the revocation of object privileges affects other users. If **user1** grants a SELECT on a table with GRANT OPTION to **user2**, and **user2** grants the SELECT on the table to **user3**, if the SELECT privilege is revoked from **user2** by **user1**, then **user3** also loses the SELECT privilege. This is a **critical** difference.

Table Privileges

Table privileges are schema object privileges specifically applicable to Data Manipulation Language (DML) operations and Data Definition Language (DDL) operations for tables.

DML Operations

As was noted earlier, privileges to **DELETE**, **INSERT**, **SELECT**, and **UPDATE** for a table or view should only be granted to a system user account or role that need to query or manipulate the table data.

INSERT and **UPDATE** privileges can be restricted for a table to specific columns.

- A selective **INSERT** causes a new row to have values inserted for columns that are specified in a privilege – all other columns store **NULL** or pre-defined default values.
- A selective **UPDATE** restricts updates only to privileged columns.

DDL Operations

The **ALTER**, **INDEX**, and **REFERENCES** privileges allow DDL operations on a table.

- Grant these privileges conservatively.
- Users attempting DDL on a table may need additional system or object schema privileges, e.g., to create a table trigger, the user

requires the **CREATE TRIGGER system** privilege as well as the **ALTER TABLE object privilege**.

View Privileges

As you've learned, a view is a ***virtual table*** that presents data from one or more tables in a database.

- Views show the structure of underlying tables and are essentially a stored query.
- Views store no actual data – the data displayed is derived from the tables (or views) upon which the view is based.
- A view can be queried.
- A view can be used to update data, providing the view is "updatable" by definition.

View Privileges include:

- **CREATE VIEW** – a system privilege to create a view in your schema.
- **CREATE ANY VIEW** – a system privilege to create a view in another schema.

- Your account must have been granted appropriate **SELECT**, **INSERT**, **UPDATE**, or **DELETE** object privileges on base objects underlying the view, or

- Been granted the **SELECT ANY TABLE**, **INSERT ANY TABLE**, **UPDATE ANY TABLE**, or **DELETE ANY TABLE** system privileges.

- To grant other users to access your view, you must have object privileges on the underlying objects with the **GRANT OPTION** clause or system privileges with the **ADMIN OPTION** clause.

To use a view, a system user account only requires appropriate privileges on the view itself – privileges on the underlying base objects are NOT required.

Procedure Privileges

EXECUTE and EXECUTE ANY PROCEDURE

The **EXECUTE** privilege is the only schema object privilege for procedures.

- This privilege applies to procedures, functions, and packages.

- Grant this privilege only to system users that will execute a procedure or compile another procedure that calls a procedure.

The **EXECUTE ANY PROCEDURE** system privilege provides the ability to execute any procedure in a database.

Roles can be used to grant privileges to users.

Definer and Invoker Rights

In order to grant EXECUTE to another user, the procedure owner must have all necessary object (or system) privileges for objects referenced by the procedure. The individual user account granting **EXECUTE** on a procedure is termed the **Definer**.

A user of a procedure requires only the EXECUTE privilege on the procedure, and does NOT require privileges on underlying objects. A user of a procedure is termed the **Invoker**.

At runtime, the privileges of the **Definer** are checked – if required privileges on referenced objects have been revoked, then neither the **Definer** or any **Invoker** granted **EXECUTE** on the procedure can execute the procedure.

Other Privileges

CREATE PROCEDURE or **CREATE ANY PROCEDURE** system privileges must be granted to a user account in order for that user to create a procedure.

To alter a procedure (manually recompile), a user must own the procedure or have the **ALTER ANY PROCEDURE** system privilege.

Procedure owners must have appropriate schema object privileges for any objects referenced in the procedure body – these must be explicitly granted and cannot be obtained through a role.

Type Privileges

Type privileges are typically system privileges for named types that include object types, VARRAYs, and nested tables. The system privileges in this area are detailed in this table.

Privilege	**Allows a user account to:**
CREATE TYPE	Create a named type in your own schema.
CREATE ANY TYPE	Create a named type in any schema.
ALTER ANY TYPE	Alter a type in any schema.
DROP ANY TYPE	Drop a named type in any schema.
EXECUTE ANY TYPE	Use and reference a named type in any schema (not obtainable through a role).

The **CONNECT** and **RESOURCE** roles are granted the **CREATE TYPE** system privilege and the **DBA** role includes all of the above privileges.

Object Privileges

The **EXECUTE** privilege permits a user account to use the type's methods. The user can use the named type to:

- Define a table.
- Define a column in a table.
- Declare a variable or parameter of the named type.

Example from Oracle Database Security Guide Part Number B10773-01 documentation:

Assume that three users exist with the **CONNECT** and **RESOURCE** roles:

- **User1**
- **User2**
- **User3**

User1 performs the following DDL in his schema:

```
CREATE TYPE Type1 AS OBJECT (
  Attribute_1 NUMBER);

CREATE TYPE Type2 AS OBJECT (
  Attribute_2 NUMBER);

GRANT EXECUTE ON Type1 TO User2;
GRANT EXECUTE ON Type2 TO User2 WITH GRANT OPTION;
```

User2 performs the following DDL in his schema:

```
CREATE TABLE Tab1 OF User1.Type1;
CREATE TYPE Type3 AS OBJECT (
  Attribute_3 User1.Type2);
CREATE TABLE Tab2 (
  Column_1 User1.Type2);
```

The following statements succeed because **User2** has EXECUTE privilege on **User1's** TYPE2 with the GRANT OPTION:

```
GRANT EXECUTE ON Type3 TO User3;
GRANT SELECT on Tab2 TO User3;
```

However, the following grant fails because **User2** does not have EXECUTE privilege on **User1's** TYPE1 with the GRANT OPTION:

```
GRANT SELECT ON Tab1 TO User3;
```

Data Dictionary Information

Displaying Schema Object Privileges

Several views provide information about object privileges. These can be queried as you have time and include:

- **DBA_TAB_PRIVS** - all object privileges granted to a user.
- **DBA_COL_PRIVS** - all privileges granted on specific columns of a table.

END OF NOTES