

CST8276: Lab 5 – Additional Perspective on Roles and Privileges

Purpose:

In last week's lab, you created a user and **granted** privileges. This week, rather than use predefined roles, we will create new roles, grant privileges to the roles and then grant the roles to particular users based on the user's requirements.

Deliverable:

Complete this document, submit it via BrightSpace, and demonstrate the results to the lab professor. This lab is worth 2 marks towards the lab/assignments component of your final grade.

Requirements:

1. **Predefined Privileges:** Oracle classifies SYSTEM privileges as actions that apply to the system or types of objects. For example, CREATE DATABASE, CREATE TABLESPACE, and CREATE TABLE. System privileges are listed and described in Table 18.1 of Oracle's Database SQL Language Reference Guide, http://docs.oracle.com/database/121/SQLRF/statements_9013.htm#SQLRF01603

Oracle classifies OBJECT privileges as actions that apply to a particular object. For example, SELECT, INSERT, UPDATE, and DELETE actions on a specific table. Additionally, the owner of an object – say I created a table named *Test* - would automatically have SELECT, INSERT, UPDATE, etc. rights on *Test*. A user can also be granted rights on an object owned by another user – we will do this later in the current lab. Object privileges are listed and described in Table 18.2.

Start SQLPlus and logon as SYS AS SYSDBA.

- a. How many predefined system privileges are listed in the system_privilege_map view: _____237_____
 - b. How many predefined object privileges are listed in the dba_tab_privs view: _____45021_____ (Hint: You won't want to *SELECT * FROM dba_tab_privs* as a large number of rows will be returned. Instead use an aggregation function to determine the number of rows in the view).
2. **Creating Roles:** The information on the next page describes privileges that certain users have on specific tables. Technically, we could grant the correct object privilege on each table on a user-by-user basis ... but that approach is inelegant, inefficient and scales poorly – what is feasible for the first 6 users is terribly problematic for the next 30,000. A more realistic (simplified) example would be BrightSpace – it has an ever-increasing number of users assigned to the *student* role and approximately 1,000 users assigned to the *instructor* role – with each role having different privileges.

CST8276: Lab 5 – Additional Perspective on Roles and Privileges

Object Privileges	Tables (Objects)			
	AUTHORS	AUTHOR_TITLES	TITLES	PUBLISHERS
SELECT	UserA, UserB, UserC, UserD, UserE	UserA, UserB, UserC, UserD, UserE	UserA, UserB, UserC, UserD, UserE, UserF	UserA, UserB, UserC, UserD, UserE
INSERT	UserA, UserB,	UserC, UserD, UserE	UserF	UserC, UserD, UserE
UPDATE	UserA, UserB,	UserC, UserD, UserE	UserF	UserC, UserD, UserE
DELETE	UserA, UserB,	UserC, UserD, UserE	UserF	UserC, UserD, UserE

- a. Take the data from the table above and re-organize it in the format below. I have done the first row for you. Doing this will help you spot the privileges/object pairings for each user more easily. (Note: S=Select, I=Insert, U=Update, and D=Delete).

	Authors				Author_Titles				Titles				Publishers			
	S	I	U	D	S	I	U	D	S	I	U	D	S	I	U	D
UserA	Y	Y	Y	Y	Y	N	N	N	Y	N	N	N	Y	N	N	N
UserB	Y	Y	Y	Y	Y	N	N	N	Y	N	N	N	Y	N	N	N
UserC	Y	N	N	N	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y
UserD	Y	N	N	N	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y
UserE	Y	N	N	N	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y
UserF	N	N	N	N	N	N	N	N	Y	Y	Y	Y	N	N	N	N

- b. What is the **minimum** number of roles (i.e., grouping by common requirements) required to assign the correct privileges to the appropriate users? → 3.
- c. List the roles (e.g., role1, role2, etc.) and the users that should be assigned to each role.

Role1 → UserA, UserB
 Role2 → UserC, UserD, UserE
 Role3 → UserF

- d. Use the template below to create 6 users named UserA, UserB, UserC, UserD, UserE and UserF, and to grant them connect privileges.

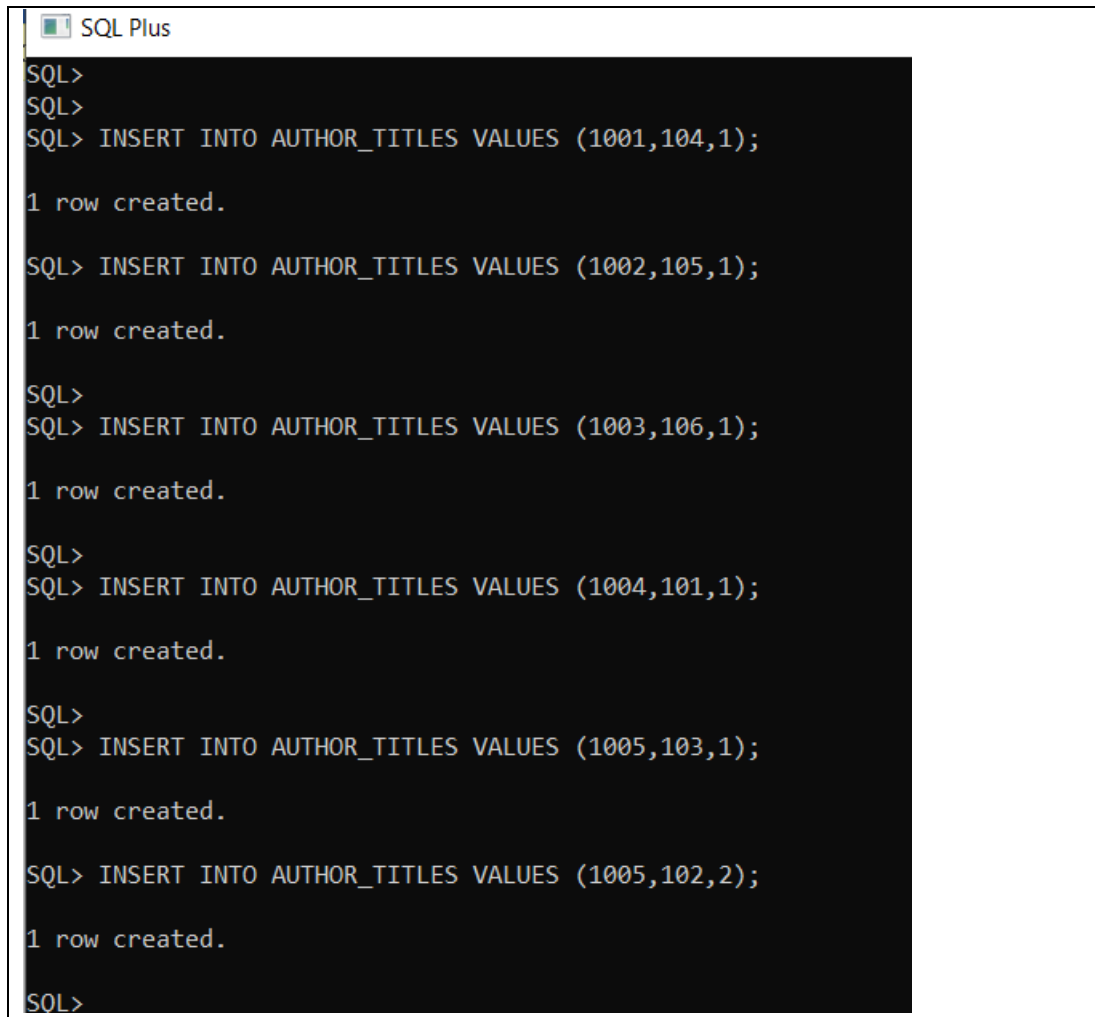
```
CREATE USER UserA IDENTIFIED BY userapswd
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp
QUOTA UNLIMITED ON users;
```

```
Grant connect to usera;
```

CST8276: Lab 5 – Additional Perspective on Roles and Privileges

Can these users create a table? No Why? because that privilege wasn't given to the user

- e. Logon to your general user account (e.g., *king*). Use the posted create tables file to create the *Authors*, *Author_Titles*, *Titles*, and *Publishers* tables. Use the posted insert file to add data to the tables.



```
SQL Plus
SQL>
SQL>
SQL> INSERT INTO AUTHOR_TITLES VALUES (1001,104,1);
1 row created.
SQL> INSERT INTO AUTHOR_TITLES VALUES (1002,105,1);
1 row created.
SQL>
SQL> INSERT INTO AUTHOR_TITLES VALUES (1003,106,1);
1 row created.
SQL>
SQL> INSERT INTO AUTHOR_TITLES VALUES (1004,101,1);
1 row created.
SQL>
SQL> INSERT INTO AUTHOR_TITLES VALUES (1005,103,1);
1 row created.
SQL> INSERT INTO AUTHOR_TITLES VALUES (1005,102,2);
1 row created.
SQL>
```

- f. Logon as SYS AS SYSDBA and enter: *CREATE ROLE Rolefname1*; (for example, *Roleking1*). Create the number of roles you determined were needed in Question 2a above. Additional information on creating roles can be found here: http://docs.oracle.com/database/121/SQLRF/statements_6014.htm#SQLRF01311

CST8276: Lab 5 – Additional Perspective on Roles and Privileges

```
SQL> CREATE ROLE Role1;
Role created.
SQL> CREATE ROLE Role2;
Role created.
S
SQL> CREATE ROLE Role3;
Role created.
SQL>
```

- g. Review the basic syntax for GRANTING an object privilege (e.g., SELECT, etc.) on a schema object (e.g., *king.authors*) to a role (e.g., *Roleking1*). Read the **Restriction on Object Privileges** sentence under **Grant_Object_Privileges** clause description as it contains important information regarding whether, for example, you can grant a select on *king.authors* and select on *king.titles* in the same grant statement. Granting these two privileges in two separate grant statements is fine.

http://docs.oracle.com/database/121/SQLRF/statements_9013.htm#SQLRF01603

Copy your GRANTs here:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.AUTHORS TO
Role1;
GRANT SELECT ON SYSTEM.SUBJECTS TO Role1;
GRANT SELECT ON SYSTEM.TITLES TO Role1;
GRANT SELECT ON SYSTEM.PUBLISHERS TO Role1;

GRANT SELECT ON SYSTEM.AUTHORS TO Role2;
GRANT SELECT ON SYSTEM.TITLES TO Role2;

GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.SUBJECTS TO
Role2;
GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.PUBLISHERS
TO Role2;

GRANT SELECT, INSERT, UPDATE, DELETE ON SYSTEM.TITLES TO
Role3;
```

CST8276: Lab 5 – Additional Perspective on Roles and Privileges

- h. Enter: *DESC role_tab_privs* for a description of the structure of the view that contains information about roles and their privileges. You may notice the first 5 columns are unnecessarily wide for our data. Use the COLUMN / FORMAT command to modify the displayed column width. Use this example as a guide:
SQL> **column role format a10**

Write a SQL query that selects all columns from the *role_tab_privs* view for all of your newly created roles and orders the results by role, table_name, and privilege. Paste the command and results here:

- i. Our next step is to grant the appropriate role or roles to the correct user(s). Granting a role to a user is very similar to granting a privilege to a role or, for that matter, granting a role to another role. Use 'g' above as a model for granting a role to a user. Paste the commands and results here:

```
SQL> GRANT ROLE1 TO SYSTEM;
Grant succeeded.
SQL> GRANT ROLE2 TO SYSTEM;
Grant succeeded.
SQL> GRANT ROLE3 TO SYSTEM;
Grant succeeded.
SQL> _
```

CST8276: Lab 5 – Additional Perspective on Roles and Privileges

Additional detailed information is available in the GRANT section of the SQL Reference Guide

http://docs.oracle.com/database/121/SQLRF/statements_9013.htm#SQLRF01603

- j. After all of the role granting to users has completed, enter the following command and then paste the query and results in the space provided below:

```
select *  
from role_tab_privs  
where role like 'ROLE%'  
order by role, table_name, privilege;
```

Paste the query and results here:

```
SQL> select *  
2  from role_tab_privs  
3  where role like 'ROLE%'  
4  order by role, table_name, privilege;  
  
ROLE  
-----  
OWNER  
-----  
TABLE_NAME  
-----  
COLUMN_NAME  
-----  
PRIVILEGE                                GRA COM  
-----  
ROLE1  
SYSTEM  
AUTHORS  
  
ROLE  
-----  
OWNER  
-----  
TABLE_NAME  
-----  
COLUMN_NAME  
-----  
PRIVILEGE                                GRA COM  
-----
```

CST8276: Lab 5 – Additional Perspective on Roles and Privileges

- k. To demonstrate that the **GRANTs** were correctly established, write the query and show the results for the following actions:

- a. UserA inserting Author ID = 60 in Authors

```
SQL> INSERT INTO SYSTEM.AUTHORS VALUES(60, 'APO', 'APO', 'APO');  
1 row created.
```

- b. UserA inserting Publisher ID = 60 in Publishers

```
SQL> INSERT INTO SYSTEM.PUBLISHERS VALUES(60, 'APO', 'APO', 'APO');  
1 row created.
```

- c. UserF selecting publisher ID and Title ID from Titles

```
SQL> SELECT PUBID, TITLEID FROM SYSTEM.TITLES;  
  
PUBID  TITLEID  
-----  
4      1001  
3      1002  
1      1003  
3      1004  
2      1005
```

You're done!