Database Security – master, 2nd year
Laboratory 4

Roles and privileges

Keywords: • Privilege (system, object)	• Role • Hierarchy of privileges

1. Recap

- In the previous lab we studied how to restrict access to computational resources. We recall:

  „*The next step, after creating user accounts and establishing storage spaces, is to impose restrictions on access to resources for users. The purpose is to ensure the efficient functioning of the database, the avoidance of the monopoly by a user over resources etc.*

  The performance parameters that are often found in these configurations refer to:

  - *estimated execution time of the statements; CPU consumption;*
  - *the degree of parallelism accepted in multi-processor systems;*
  - *the number of open sessions per user; the idle time."*

- In this lab we will study another type of restriction of the database users' activity: through privileges and roles.

2. Privileges

- **A privilege** is a right to execute a particular type of SQL statement or to access another user's objects. There are 2 categories of privileges: system and object privileges, which are summarized in Table 1.

- The granting of a privilege and the withdrawal of a privilege to / from users, respectively, is done according to the syntax:

```
GRANT privilege_1,privilege_2,…,privilege_n
[ON object] TO user
[WITH GRANT OPTION];

REVOKE privilege_1,privilege_2,…,privilege_n
[ON object] FROM user;
```

- At any time a user can find out the privileges of the current session through the query:

```
SELECT * FROM session_privs;
```

```
SQL>  select * from session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
CREATE TABLE
CREATE ANY TABLE
```

- Connected as SYS/SYSDBA we can find out the privileges of any user through the query:

```
SELECT substr(grantee,1,20) grantee, owner,
       substr(table_name,1,15) table_name,
       grantor,privilege
FROM DBA_TAB_PRIVS
WHERE grantee like '%ELEARN%';
```

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE
ELEARN_PROFESOR1	ELEARN_APP_ADMIN	CURS	ELEARN_APP_ADMIN	UPDATE
ELEARN_PROFESOR1	ELEARN_APP_ADMIN	CURS	ELEARN_APP_ADMIN	SELECT
ELEARN_PROFESOR1	ELEARN_APP_ADMIN	CURS	ELEARN_APP_ADMIN	INSERT

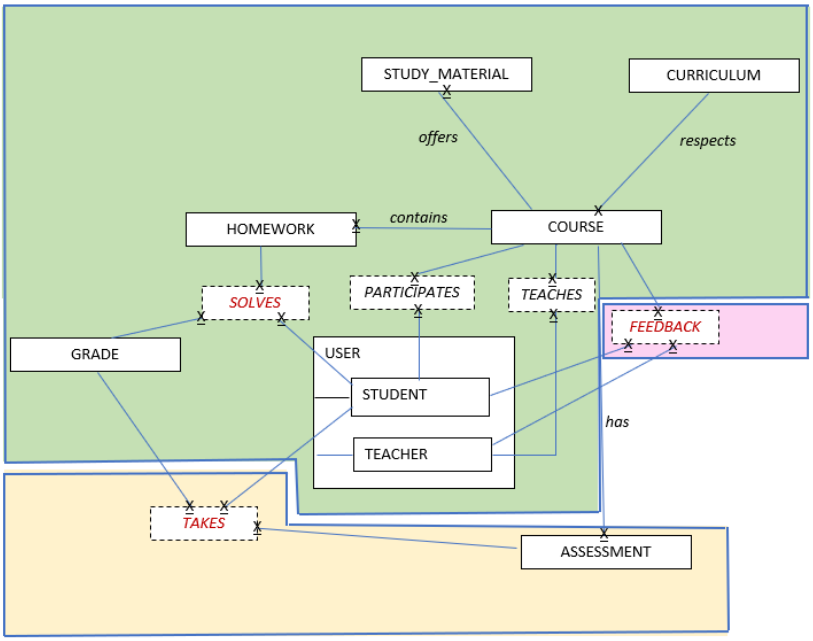
Remark: Although SYS granted it the privilege, the owner of the object, ELEARN_APP_ADMIN, appears as the grantor.

- Connected as SYS/SYSDBA we can find out the privileges of all users on a certain object of the database with the query:

```
SELECT substr(grantee,1,15) grantee, owner,
       substr(table_name,1,15) table_name, grantor,privilege
FROM DBA_TAB_PRIVS
WHERE table_name='CURS';
```

- Privileges are classified in: system privileges and object privileges (see the table in the following section).
  - System privileges can be granted with the clause WITH ADMIN OPTION. Object privileges can be granted with the clause WITH GRANT OPTION. Both options allow their holders to grant the privilege further to other users.

- Consider the example of the e-learning application started in laboratory 3, from which we recall the diagram:



User	**Password change**	**Objects in the user schema, according to the application's design**
ELEARN_APP_ADMIN	`alter user ELEARN_APP_ADMIN identified by 12345;`	STUDY_MATERIAL, CURRICULUM, HOMEWORK, COURSE, GRADE, USER, STUDENT, TEACHER, SOLVES, PARTICIPATES, TEACHES
ELEARN_student1	`alter user ELEARN_student1 identified by 12345;`	
ELEARN_student2		
ELEARN_student3		
ELEARN_student4		
ELEARN_student5		
ELEARN_student6		
ELEARN_student7		
ELEARN_student8		
ELEARN_student9		
ELEARN_student10		
ELEARN_profesor1	`alter user ELEARN_professor1 identified by 12345;`	FEEDBACK
ELEARN_profesor2		FEEDBACK
ELEARN_asistent3	`alter user ELEARN_assistant3 identified by 12345;`	FEEDBACK
ELEARN_GUEST		
OPS$MM-33C58500149B\ELEARN_CAT		ASSESSMENT, TAKES

3. Types of privileges

Table 1. Types of privileges, with comments and examples (connection *sys as sysdba*)

Privilege type	Comments	Examples of privileges	Exemples related to our data model
1. System privileges	1.1. Allow the user to connect to the database	CREATE SESSION	`GRANT CREATE SESSION TO ELEARN_APP_ADMIN;`

	1.2. Allow the user to create any object that has a certain type and belongs to his own schema (over which he is the owner)	CREATE TABLE
		CREATE SEQUENCE
		CREATE VIEW
			• The user ELEARN_APP_ADMIN tries to create the table COURSE in his own schema:
			`CREATE TABLE COURSE`
			`(id number(6) primary key,`
			`course_name varchar2(30) NOT NULL,`
			`year_of_study number(1) NOT NULL, semester number(1) NOT NULL,`
			`nb_credits number(1) NOT NULL,`
			`assessment_type VARCHAR2(10) DEFAULT 'EXAM',`
			`nb_course_hours number(2) DEFAULT 28,`
			`nb_lab_hours number(2) DEFAULT 14,`
			`nb_seminar_hours number(2) DEFAULT 0);`
			`CREATE TABLE CURS`
			`*`
			`ERROR at line 1:`
			`ORA-01031: insufficient privileges`
			• SYS (AS SYSDBA) grants the privilege to create tables to the user ELEARN_APP_ADMIN in his schema:
			`GRANT CREATE TABLE TO ELEARN_APP_ADMIN;`
			• The user ELEARN_APP_ADMIN tries again to create the table COURSE in his own schema:
			`Table created.`
			• From now on, the user ELEARN_APP_ADMIN is the owner of the table COURSE and can successfully perform DDL and DML operations on it:
			`ALTER TABLE COURSE drop column nb_seminar_hours;`
			`DROP TABLE COURSE ; -- then re-create it`
			`INSERT INTO COURSE VALUES (1,'DATABASE SECURITY',`
			`6,1,5,'E',28,14,0);`
			• Note that we cannot revoke the privileges on a table from the table's owner.

	1.3. Allow the user to perform a specific DDL or DML operation on any object, that has a certain type (table, trigger etc.), in any user's schema
	
	`select name from system_privilege_map where name like '%ANY%' order by name;`
		CREATE ANY TABLE
		ALTER ANY TABLE
		DROP ANY TABLE
		DROP ANY VIEW
		DROP ANY TRIGGER
		SELECT ANY TABLE
		INSERT ANY TABLE
		UPDATE ANY TABLE
		DELETE ANY TABLE
		EXECUTE ANY PROCEDURE
			• The user ELEARN_APP_ADMIN tries to create a table in the schema of the user ELEARN_professor1:
			`CREATE TABLE ELEARN_professor1.FEEDBACK ( id number(6) primary key, message varchar2(200), feedback_date date date);`
			`CREATE TABLE ELEARN_profesor1.FEEDBACK`
			`*`
			`ERROR at line 1:`
			`ORA-01031: insufficient privileges`
			• SYS (AS SYSDBA) grants to the user ELEARN_APP_ADMIN the privilege to create tables in any user's schema:
			`GRANT CREATE ANY TABLE TO ELEARN_APP_ADMIN;`
			• The user ELEARN_APP_ADMIN tries again to create the table FEEDBACK in the user ELEARN_professor1's schema:
			=> AGAIN, "INSUFFICIENT PRIVILEGES" ERROR; as there is a primary key
			=> the privilege CREATE ANY INDEX is also needed
			• SYS AS SYSDBA grants to the user ELEARN_APP_ADMIN the privilege to create tables and indexes in any user's schema:
			`GRANT CREATE ANY TABLE TO ELEARN_APP_ADMIN;`
			`GRANT CREATE ANY INDEX TO ELEARN_APP_ADMIN;`
			• The user ELEARN_APP_ADMIN tries again to create the table FEEDBACK in the schema of the user ELEARN_professor1:
			`SQL> CREATE TABLE ELEARN_profesor1.FEEDBACK`
			`Table created.`
			• Find out who owns the table ELEARN_professor1.FEEDBACK:
			`select owner, object_name from all_objects where owner like '%ELEARN%';`
			`OWNER OBJECT_NAME`
			`ELEARN_APP_ADMIN CURS`
			`ELEARN_APP_ADMIN SYS_C0011548`
			`ELEARN_PROFESOR1 FEEDBACK`
			`ELEARN_PROFESOR1 SYS_C0011559`
			• Conclusion: the owner will be the user in whose schema the object is created, no matter who creates it.

			• Find out: between ELEARN_professor1 and ELEARN_APP_ADMIN, which operations are successfully completed for each? (At this time, the user ELEARN_professor1 has only the CREATE SESSION privilege): `ALTER TABLE ELEARN_professor1.FEEDBACK drop column feedback_date;` --succeeds only for ELEARN_professor1 `DROP TABLE ELEARN_professor1.FEEDBACK ;` -- succeeds only for ELEARN_professor1 -- then, the table is re-created by ELEARN_APP_ADMIN `INSERT INTO ELEARN_professor1.FEEDBACK VALUES (1, 'very interesting and useful subject', SYSDATE);` -- succeeds only for ELEARN_professor1 • Conclusion: even if an user X does not have explicit privileges for DDL, DML operations not even for its own schema, if another user Y creates an object in his schema, then the user X will be able to successfully perform DDL and DML operations on that object.

2. Privileges on schema's objects	They have effect within a certain object of a database schema.
	
	Applied on a synonym of an object, they have the same effect as being applied directly on the object.
	
	The privilege ALL applied on an object of a schema grants full rights on it.
		2.1) Table privileges
		• DML
		SELECT ON schema.table
		INSERT ON schema.table
		UPDATE ON schema.table
		DELETE ON schema.table
		
		• DDL
		ALTER ON schema.table
		INDEX ON schema.table
			• Based on the ***entity-user matrix***, the following privileges are given on the schema objects: select, insert and update on the ELEARN_APP_ADMIN.COURSE table to the ELEARN_professor1 user.
			`GRANT SELECT,INSERT,UPDATE ON ELEARN_APP_ADMIN.COURSE TO ELEARN_professor1;`
			• Testing, logged in as ELEARN_professor1:
			`INSERT INTO ELEARN_APP_ADMIN.COURSE VALUES (2,'NETWORKING',3, 2,5,'E',28,28,0);`
			`SQL> INSERT INTO ELEARN_APP_ADMIN.CURS VALUES`
			`1 row created.`
			• We saw above (1.3) that the command executed by ELEARN_APP_ADMIN fails:
			`ALTER TABLE ELEARN_professor1.FEEDBACK drop column feedback_date;`
			• The user is granted this privilege (by SYS/ AS SYSDBA):
			`GRANT ALTER ON ELEARN_professor1.FEEDBACK TO ELEARN_APP_ADMIN;`
			• Testing: First, ELEARN_professor1 executes the commands DELETE FROM FEEDBACK (otherwise => error "resource busy" as we cannot remove a column having data) and COMMIT. Then, ELEARN_APP_ADMIN executes again the command ALTER TABLE, successfully. Also, ELEARN_APP_ADMIN executes the following commands, successfully:

			ALTER TABLE ELEARN_professor1.FEEDBACK add student_id NUMBER(4); ALTER TABLE ELEARN_professor1.FEEDBACK add feedback_date date;
		2.2) View privileges SELECT ON schema.viz INSERT ON schema.viz UPDATE ON schema.viz DELETE ON schema.viz	We propose the following scenario as an application: • SYS AS SYSDBA grants the following privileges to the user ELEARN_APP_ADMIN: GRANT CREATE ANY TABLE TO ELEARN_APP_ADMIN; GRANT CREATE ANY INDEX TO ELEARN_APP_ADMIN; • The user ELEARN_APP_ADMIN creates the tables: CREATE TABLE ELEARN_professor1.FEEDBACK (id number(6) primary key, message varchar2(200), student_id number(4), feedback_date date); CREATE TABLE ELEARN_profesor2.FEEDBACK (id number(6) primary key, message varchar2(200), student_id number(4), feedback_date date); CREATE TABLE ELEARN_asistent3.FEEDBACK (id number(6) primary key, message varchar2(200), student_id number(4), feedback_date date); • A student can insert feedback for teachers who taught them. The designer of the e-learning application provided a view through which the corresponding insertions in the base tables should be made, by means of a trigger *instead of*. Thus, ELEARN_APP_ADMIN wants to create this view (FEEDB_VIEW): CREATE OR REPLACE VIEW FEEDB_VIEW AS SELECT MESSAGE,STUDENT_ID,'PROF1' AS prof FROM ELEARN_professor1.FEEDBACK UNION SELECT MESSAGE,STUDENT_ID,'PROF2' AS prof FROM ELEARN_professor2.FEEDBACK UNION SELECT MESSAGE,STUDENT_ID,'ASIST3' AS prof FROM ELEARN_assistant3.FEEDBACK; • Initially, ELEARN_APP_ADMIN will get the “insufficient privileges” error.

- To resolve this issue, SYS / AS SYSDBA grants to the user the following privileges:
  - **System privilege for view creation in his own schema (1.1):**

```
GRANT CREATE VIEW TO ELEARN_APP_ADMIN ;
```

==> It is not enough, he still gets the "insufficient privileges" error.

  - **Objet privilege for selection on each of the tables involved in the view's query, with grant option:**

```
GRANT SELECT ON ELEARN_professor1.FEEDBACK TO
ELEARN_APP_ADMIN  WITH GRANT OPTION;
GRANT SELECT ON ELEARN_professor2.FEEDBACK TO
ELEARN_APP_ADMIN  WITH GRANT OPTION;
GRANT SELECT ON ELEARN_assistant3.FEEDBACK TO
ELEARN_APP_ADMIN WITH GRANT OPTION;
```

==> now the view is successfully created by ELEARN_APP_ADMIN .

- We grant the privilege to query the view to a student:

```
GRANT SELECT ON ELEARN_APP_ADMIN.FEEDB_VIEW TO
ELEARN_student1;
```

- The student queries the view successfully:

```
SELECT * FROM ELEARN_APP_ADMIN.FEEDB_VIEW WHERE student_id=1;
```

- Regarding the insertion of data in the base tables by means of the view: since the view contains the UNION operator ==> the view is complex ==> data cannot be inserted directly through the view, but an INSTEAD OF trigger is needed.
- It is necessary for the user ELEARN_APP_ADMIN to be able to create triggers in his own schema. SYS AS SYSDBA grants him the privilege:

```
GRANT CREATE TRIGGER TO ELEARN_APP_ADMIN;
```

- In addition, ELEARN_APP_ADMIN should receive privileges to insert into the view's base tables, with the clause *with grant option*:

```
GRANT INSERT ON ELEARN_professor1.FEEDBACK TO ELEARN_APP_ADMIN
WITH GRANT OPTION;
GRANT INSERT ON ELEARN_professor2.FEEDBACK TO ELEARN_APP_ADMIN
WITH GRANT OPTION;
GRANT INSERT ON ELEARN_assistant3.FEEDBACK TO ELEARN_APP_ADMIN
WITH GRANT OPTION;
```

- ELEARN_APP_ADMIN creates an INSTEAD OF trigger as follows:

```
CREATE OR REPLACE TRIGGER TR_FEEDB
INSTEAD OF INSERT ON FEEDB_VIEW
FOR EACH ROW
BEGIN
   IF :NEW.PROF='PROF1' THEN
      INSERT INTO ELEARN_professor1.FEEDBACK
      VALUES(SYSDATE - TO_DATE('2000-01-01', 'yyyy-mm-dd'),
             :NEW.MESSAGE,101,SYSDATE);
   END IF;
   IF :NEW.PROF='PROF2' THEN
      INSERT INTO ELEARN_professor2.FEEDBACK
      VALUES(SYSDATE - TO_DATE('2000-01-01', 'yyyy-mm-dd'),
             :NEW.MESSAGE,101,SYSDATE);
   END IF;
   IF :NEW.PROF='ASIST3' THEN
      INSERT INTO ELEARN_assistant3.FEEDBACK
      VALUES(SYSDATE - TO_DATE('2000-01-01', 'yyyy-mm-dd'),
             :NEW.MESSAGE,101,SYSDATE);
   END IF;
END;
/
Trigger created.
```

- Now SYS (AS SYSDBA) grants the insert privilege on the view to the student ELEARN_student1:

```
GRANT INSERT ON ELEARN_APP_ADMIN.FEEDB_VIEW
TO ELEARN_student1;
```

- The student executes the following insertion:

```
INSERT INTO ELEARN_APP_ADMIN.FEEDB_VIEW VALUES('AN INTERESTING
COURSE',101,'PROF1');
SQL> INSERT INTO ELEARN_APP_ADMIN.VIZ_FEEDB VALUES

1 row created.

COMMIT; -- mandatory !
```

- The professor ELEARN_professor1 checks if he has received any feedback:

```
SELECT ID, SUBSTR(MESSAGE,1,20) MESSAGE, STUDENT_ID,FEEDBACK_DATE
FROM FEEDBACK;
SQL> SELECT ID, SUBSTR(MESAJ,1,20) MESAJ, COD_STUDENT,TIMP FROM FEEDBACK;

        ID MESAJ                COD_STUDENT TIMP
---------- -------------------- ----------- ---------
      4620 UN CURS INTERESANT           101 24-AUG-12
```

		2.3)
		Procedure privileges
		
		EXECUTE ON schema.procedure
			There are two situations:
			• **Situation 1)**, similar to the view and the trigger created before, the procedure's creator receives the right to create a procedure in his own schema and, in addition, he receives the appropriate privileges on the objects that are referred within the procedure, with the clause WITH GRANT OPTION.
			• In this case, the caller needs only the *execute* privilege on the procedure.
			• Thus, ELEARN_APP_ADMIN wants to create the procedure DELETE_SPAM which shoud delete the spam comments from the professor's FEEDBACK table. The procedure gets minimum 1, maximum 3 inputs (the spam keywords that should be searched for in the messages):
			`CREATE OR REPLACE PROCEDURE DELETE_SPAM(key1 VARCHAR2, key2 VARCHAR2 default 'advertisement', key3 VARCHAR2 default 'publicity') AS`
			`BEGIN`
			`DELETE FROM ELEARN_professor1.FEEDBACK`
			`WHERE MESSAGE LIKE '%'||key1||'%' OR MESAGE LIKE '%'||key2||'%'`
			`OR MESAGE LIKE '%'||key3||'%';`
			`DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' spam mesages have been deleted from the professor 1''s table');`
			`COMMIT;`
			`END;`
			`/`
			CREATE OR REPLACE PROCEDURE DELETE_SPAM(
			licitate') DELETE_SPAM AS
			*****
			ERROR at line 1:
			ORA-01031: insufficient privileges
			• To solve this issue, SYS/AS SYSDBA grants to ELEARN_APP_ADMIN the following privileges:
			***** The privilege to create procedures in his own schema:
			`GRANT CREATE PROCEDURE TO ELEARN_APP_ADMIN;`
			***** Privileges on the objects referred by the procedure, with GRANT OPTION:
			`GRANT DELETE ON ELEARN_professor1.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			• Now , ELEARN_APP_ADMIN will succeed to create the procedure.
			• Furthermore, the user ELEARN_APP_ADMIN can execute the procedure successfully:

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC DELETE_SPAM('AVANTAJOS');
Au fost sterse:0 mesaje de tip spam din tabela profesorului 1

PL/SQL procedure successfully completed.
```

- We assume that the application's designer has established that this task of deleting spam messages can also be performed by an assistant. When he tries to execute it, he receives an error:

```
SQL> EXEC ELEARN_APP_ADMIN.DELETE_SPAM('AVANTAJOS');
BEGIN ELEARN_APP_ADMIN.DELETE_SPAM('AVANTAJOS'); END;

      *
ERROR at line 1:
ORA-06550: line 1, column 7:
PLS-00201: identifier 'ELEARN_APP_ADMIN.DELETE_SPAM' must be declared
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
```

- Therefore, the user ELEARN_assistant3 gets the following privileges from SYS (AS SYSDBA):

```
GRANT EXECUTE ON ELEARN_APP_ADMIN.DELETE_SPAM TO
ELEARN_assistant3;
```

- Now, the assistant will also be able to execute the procedure successfully.

```
EXEC ELEARN_APP_ADMIN.DELETE_SPAM('AVANTAJOS');
```

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC ELEARN_APP_ADMIN.DELETE_SPAM('AVANTAJOS');
Au fost sterse:0 mesaje de tip spam din tabela profesorului 1

PL/SQL procedure successfully completed.
```

- **Situation 2)** If the creator of the procedure has appropriate privileges on the objects which are referred within the procedure, but without the grant option, THEN the caller will need to get the respective rights himself. Otherwise, he will not be able to successfully execute the procedure.

4. Recap of the situations encountered in the examples

User X creates a view object (trigger, procedure -)

	In X's own schema			In another user (Y)'s schema		
	Accesses objects in X's own schema	Accesses objects in the Y's schema (select Y.D, insert Y.D)		Accesses objects in X's own schema	Accesses objects in the Y's schema (select Y.D, insert Y.D)	
What privileges are needed by X?	CREATE VIEW	CREATE VIEW SELECT ON Y.D INSERT ON Y.D	CREATE VIEW SELECT ON Y.D WITH GRANT OPTION INSERT ON Y.D WITH GRANT OPTION	CREATE ANY VIEW	CREATE ANY VIEW SELECT ON Y.D INSERT ON Y.D	CREATE ANY VIEW SELECT ON Y.D WITH GRANT OPTION INSERT ON Y.D WITH GRANT OPTION
What privileges are needed by a caller Z?	SELECT ON view INSERT ON view	SELECT ON view INSERT ON view SELECT ON Y.D INSERT ON Y.D	SELECT ON view INSERT ON view	SELECT ON view INSERT ON view	SELECT ON view INSERT ON view SELECT ON Y.D INSERT ON Y.D	SELECT ON view INSERT ON view

5. Roles

- **Roles** are containers for privileges, so that they are easier to manage: when a user receives a role, by default he receives all the privileges contained in that role.
- There are predefined roles in Oracle, for example:

Role	Privileges contained in the role
CONNECT	CREATE VIEW CREATE TABLE ALTER SESSION CREATE CLUSTER CREATE SESSION CREATE SYNONYM CREATE SEQUENCE CREATE DATABASE LINK
RESOURCE	CREATE TYPE CREATE TABLE CREATE CLUSTER CREATE TRIGGER CREATE OPERATOR CREATE SEQUENCE CREATE INDEXTYPE CREATE PROCEDURE
DBA	Includes all privileges, with option to administer them (option to be granted further)

Remark: SYSDBA is a special case of role, similar to DBA
Remember! Do NOT confuse SYS, which is a user, with SYSDBA, which is a role.

- **Syntax** *:*
  - Creating a role:
    ```
    CREATE ROLE role_name;
    ```
  - Granting a role to a user:
    ```
    GRANT numerol TO utilizator [WITH ADMIN OPTION];
    ```
  - Including new privileges in the created role. These will be taken over by the role's users by default (if there are no contradictions – see the role hierarchy in the next chapter):
    ```
    GRANT privilege_1,privilege_2,...,privilege_n [ON object]
    TO role_name;
    ```
- Finding out the roles of the users of the e-learning application can be done through the command:

```
SELECT * FROM DBA_role_privs WHERE grantee like '%ELEARN%';
```

- The use of roles has the advantage of easier management of privileges, but also has certain disadvantages:
  - In the procedures the roles are inhibited, they have no effect. Thus, the necessary privilege will have to be granted individually and directly to the user, not by role;

- How many roles can a user have simultaneously? Answer: zero, one or more.

Example:

```
CREATE ROLE select_all;
GRANT SELECT ANY TABLE TO select_all;

CREATE ROLE update_all;
GRANT UPDATE ANY TABLE TO update_all;


GRANT select_all TO ELEARN_APP_ADMIN;
GRANT update_all TO ELEARN_APP_ADMIN;

SELECT * FROM DBA_role_privs WHERE grantee like '%ELEARN%';
```

```
SQL> SELECT * FROM DBA_role_privs WHERE grantee like '%ELEARN%';

GRANTEE                        GRANTED_ROLE                   ADM DEF
------------------------------ ------------------------------ --- ---
ELEARN_APP_ADMIN               SELECT_TOT                     NO  YES
OPS$MM-33C58500149B\ELEARN_CAT CONNECT                        NO  YES
ELEARN_APP_ADMIN               UPDATE_TOT                     NO  YES
```

6. Hierarchy of role and privilege priorities

- There are rules for aggregating and prioritizing a user's privileges.
- Privileges and roles can be seen as ways to give certain rights, but also to impose certain restrictions. This is done through the GRANT and REVOKE mechanism for privileges and roles.

Recap: ELEARN_APP_ADMIN, as owner of the table SOLVES, executes the statements in the following table:

The user ELEARN_student1 has NO privileges on the table SOLVES				
SELECT PRIVILEGE GRANTED DIRECTLY TO THE USER	**SELECT PRIVILEGE GRANTED TO A ROLE**	**SELECT PRIVILEGE GRANTED TO A ROLE**	**SELECT PRIVILEGE GRANTED TO A ROLE**	**SELECT PRIVILEGE GRANTED DIRECTLY TO THE USER**
GRANT SELECT ON SOLVES TO ELEARN_student1;	CREATE ROLE role_stud; GRANT SELECT ON SOLVES TO role_stud;	CREATE ROLE role_stud; GRANT SELECT ON SOLVES TO role_stud;	CREATE ROLE role_stud; GRANT SELECT ON SOLVES TO role_stud;	GRANT SELECT ON SOLVES TO ELEARN_student1;
	GRANT THE ROLE TO THE USER	**GRANT THE ROLE TO THE USER**	**GRANT THE ROLE TO THE USER**	**SELECT PRIVILEGE GRANTED TO THE ROLE**
	GRANT role_stud TO ELEARN_student1;	GRANT role_stud TO ELEARN_student1;	GRANT role_stud TO ELEARN_student1;	CREATE ROLE role_stud; GRANT SELECT ON SOLVES TO role_stud;
		SELECT PRIVILEGE REVOKED DIRECTLY FROM THE USER	**SELECT PRIVILEGE REVOKED FROM THE ROLE**	**GRANT THE ROLE TO THE USER**
		REVOKE SELECT ON SOLVES FROM ELEARN_student1;	REVOKE SELECT ON SOLVES FROM role_stud;	GRANT role_stud TO ELEARN_student1;
				SELECT PRIVILEGE REVOKED FROM THE ROLE
				REVOKE SELECT ON SOLVES FROM role_stud;
SUCCES	**SUCCES**	**Error**	**FAILURE**	**SUCCES !**

Table 2

Remarks:

- A privilege granted directly to the user remains available even if it is revoked from a user's role to which this privilege previously belonged to.
- The owner of an object also has all the privileges over it, with the ADMIN option. No one can ever revoke a privilege on an object in his own schema.
- The granularity of privileges granting must be respected "in mirror" when revoking them:

```
GRANT CREATE ANY TABLE TO ELEARN_assistant3;
    ⟶ REVOKE CREATE ANY TABLE FROM ELEARN_assistant3; -- correct
    ⟶ REVOKE CREATE TABLE FROM ELEARN_assistant3; -- incorrect
```

- If a user receives a privilege only through a role, not directly, then that privilege cannot be revoked directly.
- Note that REVOKE can only be given at the level of the whole table, not at the level of individual columns.

Example*:*

```
GRANT UPDATE(deadline) ON ELEARN_APP_ADMIN.HOMEWORK TO
ELEARN_assistant3;
REVOKE UPDATE ON ELEARN_APP_ADMIN.HOMEWORK FROM ELEARN_assistant3;
```

7. Exercises

Construction of the entity-user matrix, resulting from the user-process and entity-process matrices.

	Full-time students	Part-time students	Professors	Assistants	Secretaries	Alumnii	App & DB admin	Gen. public
CURRICULUM	S	S	S	S	S	S	S	S
COURSE	S	S	I,U,S	S	S	S	I,U,S	S
STUDY_MATERIAL	S	S	I,U,D,S	I,U,D,S				
HOMEWORK	S	S	I,U,S	I,U,S				
GRADE	S	S	S	S	S			
USER							I,U	
STUDENT	S	S	S	S	S		S,I,U	
TEACHER	S	S					I,U	
ASSESSMENT	S	S	I,U,S		I,U,S			
PARTICIPATES	S	S	I,U	I,U	S		I,U,S	
TEACHES	S	S	I,U				I,U	
SOLVES	I,U	I,U	U	U				
TAKES	S	S	I,U		S			
FEEDBACK	I,U,D	I,U,D	S	S				

Legend: I= Insert , U= update , D= delete, S= select

1. Use three different ways to give the teacher users the right to get information about the columns of the *HOMEWORK* table.

Hint:

- privileges on the schema's objects granted on the table directly to users;
- view privileges granted directly to users, the view being in the admin's schema;
- role that includes privileges on schema's objects.

2. Use three different ways to give the teacher users the right to update the homework's' deadlines (column of the table *HOMEWORK*) through the application, without being able to update the rest of the homework's information.

Hint:

- privileges on the schema's objects granted on the table directly to users;
- view privileges granted directly to users, the view being in the admin's schema;
- role that includes privileges on schema's objects.

3. Create a procedure (*PROC_MARKING*) that allows teachers to mark homework. The procedure will belong to the admin's schema but will be called by professors and assistants. The procedure will receive, as input parameters, the student code, the homework code, the code of the teacher who checked the homework and the awarded grade. In the background, the procedure will verify that the homework belongs to the indicated student and that it is not already graded.

4. Create a privileges context at the level of the student users that is repeatable to any student in the system. The context will differentiate between 3rd year students (undergraduate terminal) and 5th year students (master terminal) who no longer send homework (strictly educational example).