Securitatea bazelor de date – master anul 2
Laborator 4

Roluri și privilegii

Cuvinte cheie: • Privilegiu (sistem, obiect)	• Rol • Ierarhia privilegiilor

1. Recapitulare

- În laboratorul anterior am studiat restricţionarea accesului la resurse computaţionale. Ne reamintim:

  „*Următorul pas, după crearea conturilor de utilizatori și stabilirea spațiilor de stocare, îl reprezintă impunerea unor limitări referitoare la accesul la resurse pentru utilizatori. Scopul este de a asigura o funcţionare performantă a bazei de date, evitarea monopolului de către un utilizator asupra resurselor ș.a.*

  Parametrii de performanță care sunt adesea regăsiți în aceste configurări se referă la:
  - *timpul de execuţie estimat al instrucţiunilor; consumul de CPU;*
  - *gradul de paralelism acceptat în sisteme multi-procesor;*
  - *numărul de sesiuni deschise per utilizator;timp de inactivitate (idle).”*

- În acest laborator vom studia un alt tip de restricţionare a activității utilizatorilor bazei de date: prin privilegii și roluri.

2. Privilegii

- **Un privilegiu** este un drept de a executa un tip particular de instrucţiune SQL sau de a accesa obiectele altui utilizator. Există 2 categorii de privilegii: sistem și obiect, ce sunt rezumate în tabelul 1.

- Acordarea unui privilegiu și, respectiv, retragerea unui privilegiu către utilizatori se realizează după sintaxa:

```
GRANT privilegiu_1,privilegiu_2,…,privilegiu_n
[ON obiect] TO utilizator
[WITH GRANT OPTION];

REVOKE privilegiu_1,privilegiu_2,…,privilegiu_n
[ON obiect] FROM utilizator;
```

- La orice moment de timp un utilizator își poate afla privilegiile din sesiunea curentă cu ajutorul interogării:

```
SELECT * FROM session_privs;
```

```
SQL>  select * from session_privs;

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

- Conectaţi ca SYS/SYSDBA putem afla privilegiile oricărui utilizator cu interogarea:

```
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
```

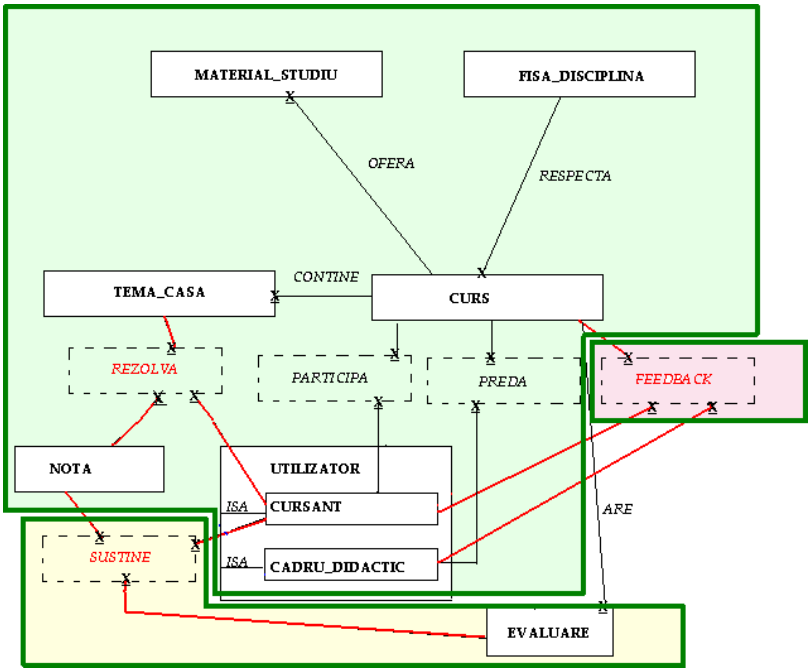
Observație: Deşi SYS i-a oferit privilegiul, ca *grantor* apare proprietarul obiectului, adică ELEARN_APP_ADMIN.

- Conectaţi ca SYS/SYSDBA putem afla privilegiile tuturor utilizatorilor pe un anumit obiect al bazei de date cu interogarea:

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

- Privilegiile se clasifică în: privilegii sistem și privilegii obiect (vezi tabelul din secțiunea următoare).
  - Privilegiile sistem pot fi acordate cu opțiunea WITH ADMIN OPTION , iar privilegiile obiect pot fi acordate cu opțiunea WITH GRANT OPTION, opțiuni ce permit deținătorilor lor să acorde la rândul lor privilegiul către alți utilizatori.

- Se consideră exemplul aplicaţiei de e-learning început în laboratorul 3, din care reamintim diagrama:



Utilizatorul	**Schimbare parolă**	**Obiectele din schema utilizatorului, conform *design*-ului aplicației**
ELEARN_APP_ADMIN	`alter user ELEARN_APP_ADMIN identified by 12345;`	MATERIAL_STUDIU, FISA_DISCIPLINA, TEMA_CASA, CURS, NOTA, UTILIZATOR, CURSANT, CADRU_DIDACTIC, REZOLVA, PARTICIPA, PREDA
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_profesor1 identified by 12345;`	FEEDBACK
ELEARN_profesor2		FEEDBACK
ELEARN_asistent3	`alter user ELEARN_asistent3 identified by 12345;`	FEEDBACK
ELEARN_GUEST		
OPS$MM-33C58500149B\ELEARN_CAT		EVALUARE, SUSTINE

3. Tipuri de privilegii

Tabelul 1. Tipuri de privilegii, cu comentarii și exemple (conectare *sys as sysdba*)

Tip de privilegiu	Comentarii	Exemple de privilegii	Exemplu real
1. Privilegii sistem	1.1. Permit utilizatorului să se conecteze la baza de date	CREATE SESSION	`GRANT CREATE SESSION TO ELEARN_APP_ADMIN;`
	1.2. Permit utilizatorului să creeze orice obiect care are un anumit tip și aparţine schemei proprii (peste care el este *owner*)	CREATE TABLE CREATE SEQUENCE CREATE VIEW	• Utilizatorul ELEARN_APP_ADMIN încearcă să creeze tabelul CURS în schema proprie: `CREATE TABLE CURS (id number(6) primary key, denumire varchar2(30) NOT NULL, an_studiu number(1) NOT NULL, semestru number(1) NOT NULL, nr_credite number(1) NOT NULL, forma_evaluare VARCHAR2(10) DEFAULT 'EXAMEN', ore_curs number(2) DEFAULT 28, ore_laborator number(2) DEFAULT 14, ore_seminar number(2) DEFAULT 0);` `CREATE TABLE CURS * ERROR at line 1: ORA-01031: insufficient privileges` • SYS (AS SYSDBA) îi acordă privilegiul de a crea tabele utilizatorului ELEARN_APP_ADMIN în schema acestuia: `GRANT CREATE TABLE TO ELEARN_APP_ADMIN;` • Utilizatorul ELEARN_APP_ADMIN încearcă din nou să creeze tabelul CURS în schema proprie: `Table created.` • Din acest moment, utilizatorul ELEARN_APP_ADMIN este *owner* (proprietar) peste tabelul CURS și poate executa cu succes operații LDD și LMD pe aceasta: `ALTER TABLE CURS drop column ore_seminar; DROP TABLE CURS ; -- apoi o recreeaza INSERT INTO CURS VALUES (1,'SECURITATEA BAZELOR DE DATE',6,1,5,'E',28,14,0);` • Reţinem că proprietarului asupra unei tabele nu îi putem revoca privilegiile asupra ei.

	1.3. Permit utilizatorului să efectueze o anumită operație LDD sau LMD asupra oricărui obiect al schemei oricărui utilizator, care are un anumit tip (tabel, trigger etc.)
	
	`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
			• Utilizatorul ELEARN_APP_ADMIN încearcă să creeze un tabel în schema utilizatorului ELEARN_profesor1:
			`CREATE TABLE ELEARN_profesor1.FEEDBACK ( id number(6) primary key, mesaj varchar2(200), timp date);`
			`CREATE TABLE ELEARN_profesor1.FEEDBACK`
			`*`
			`ERROR at line 1:`
			`ORA-01031: insufficient privileges`
			• SYS (AS SYSDBA) îi acordă privilegiul de a crea tabele utilizatorului ELEARN_APP_ADMIN în schema oricărui utilizator:
			`GRANT CREATE ANY TABLE TO ELEARN_APP_ADMIN;`
			• Utilizatorul ELEARN_APP_ADMIN incearca din nou sa creeze tabela FEEDBACK in schema utilizatorului ELEARN_profesor1:
			=> TOT EROARE DE "INSUFFICIENT PRIVILEGES"; pentru că avem un primary key => are nevoie și de CREATE ANY INDEX
			• SYS AS SYSDBA îi acordă privilegiul de a crea tabele și indecși utilizatorului ELEARN_APP_ADMIN în schema oricărui utilizator:
			`GRANT CREATE ANY TABLE TO ELEARN_APP_ADMIN;`
			`GRANT CREATE ANY INDEX TO ELEARN_APP_ADMIN;`
			• Utilizatorul ELEARN_APP_ADMIN încearcă din nou să creeze tabelul FEEDBACK în schema utilizatorului ELEARN_profesor1:
			`SQL> CREATE TABLE ELEARN_profesor1.FEEDBACK`
			`Table created.`
			• De cercetat cine apare a fi *owner* al tabelului ELEARN_profesor1.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`
			• Concluzia: proprietar va fi *user*-ul în schema căruia este creat obiectul, indiferent cine îl creează.

<table>
<tr><td></td><td></td><td></td><td>

- De cercetat: dintre ELEARN_profesor1 și ELEARN_APP_ADMIN, ce operații sunt finalizate cu succes în cazul fiecăruia? (În acest moment utilizatorul ELEARN_profesor1 are doar privilegiul CREATE SESSION):

<code>ALTER TABLE ELEARN_profesor1.FEEDBACK drop column timp;</code> --îi reușește doar lui ELEARN_profesor1

<code>DROP TABLE ELEARN_profesor1.FEEDBACK ;</code> -- îi reușește doar lui ELEARN_profesor1 -- apoi tabelul este recreat de ELEARN_APP_ADMIN

<code>INSERT INTO ELEARN_profesor1.FEEDBACK VALUES (1, 'materia este foarte interesanta si utila', SYSDATE);</code> -- îi reușește doar lui ELEARN_profesor1

- Concluzia: chiar dacă un utilizator X nu are privilegii explicite pentru operații LDD, LMD nici măcar pentru schema sa proprie, dacă alt utilizator Y îi creează un obiect în schemă, atunci asupra obiectului respectiv utilizatorul X va putea executa cu succes operații LMD și LDD.
</td></tr>
<tr><td>2. Privilegii asupra obiectelor schemei</td><td>Au efect în cadrul unui anumit obiect al unei scheme a bazei de date.

Aplicate asupra sinonimului unui obiect au acelaşi efect cu a fi aplicate direct pe obiect.

Privilegiul ALL asupra unui obiect dintr-o schemă acordă drepturi depline asupra lui.</td><td>2.1) Privilegii tabel

- LMD

SELECT ON schema.tabel
INSERT ON schema.tabel
UPDATE ON schema.tabel
DELETE ON schema.tabel

- LDD

ALTER ON schema.tabel
INDEX ON schema.tabel</td><td>

- Pe baza ***matricii entitate –utilizator*** , se dau următoarele privilegii asupra obiectelor schemei: select, insert și update pe tabelul ELEARN_APP_ADMIN.CURS către utilizatorul ELEARN_profesor1.

<code>GRANT SELECT,INSERT,UPDATE ON ELEARN_APP_ADMIN.CURS TO ELEARN_profesor1;</code>

- Testăm, conectați ca ELEARN_profesor1:

<code>INSERT INTO ELEARN_APP_ADMIN.CURS VALUES (2,'RETELE DE CALCULATOARE',3,2,5,'E',28,28,0);</code>

<code>SQL> INSERT INTO ELEARN_APP_ADMIN.CURS VALUES</code>

<code>1 row created.</code>

- Am văzut mai sus (pct 1.3) că eșuează comanda dată de ELEARN_APP_ADMIN:

<code>ALTER TABLE ELEARN_profesor1.FEEDBACK drop column timp;</code>

- Îi acordăm acest privilegiu (de către SYS/ AS SYSDBA):

<code>GRANT ALTER ON ELEARN_profesor1.FEEDBACK TO ELEARN_APP_ADMIN;</code>

- Testăm:

Mai întâi, ELEARN_profesor1 execută comenzile DELETE FROM FEEDBACK (altfel => eroare de “resource busy” pentru ca nu pot şterge o coloană care are date) și COMMIT
Apoi, ELEARN_APP_ADMIN execută din nou comanda ALTER TABLE, cu succes.
Apoi, ELEARN_APP_ADMIN execută următoarele comenzi, cu succes:
</td></tr>
</table>

			ALTER TABLE ELEARN_profesor1.FEEDBACK add cod_student NUMBER(4); ALTER TABLE ELEARN_profesor1.FEEDBACK add timp date;

		2.2) Privilegii vizualizare
		
		SELECT ON schema.viz
		
		INSERT ON schema.viz
		UPDATE ON schema.viz
		DELETE ON schema.viz
			Propunem următorul scenariu drept aplicație:
			• SYS AS SYSDBA acorda privilegiile următoare utilizatorului ELEARN_APP_ADMIN:
			`GRANT CREATE ANY TABLE TO ELEARN_APP_ADMIN;`
			`GRANT CREATE ANY INDEX TO ELEARN_APP_ADMIN;`
			• Utilizatorul ELEARN_APP_ADMIN creează tabelele:
			`CREATE TABLE ELEARN_profesor1.FEEDBACK`
			`( id number(6) primary key,`
			`mesaj varchar2(200), cod_student number(4),`
			`timp date);`
			`CREATE TABLE ELEARN_profesor2.FEEDBACK`
			`( id number(6) primary key,`
			`mesaj varchar2(200), cod_student number(4),`
			`timp date);`
			`CREATE TABLE ELEARN_asistent3.FEEDBACK`
			`( id number(6) primary key,`
			`mesaj varchar2(200), cod_student number(4),`
			`timp date);`
			• Un student poate insera *feedback* pentru profesorii care i-au predat la clasă. Proiectantul aplicației de e-learning a prevăzut o vizualizare prin intermediul căreia să se realizeze inserările corespunzătoare în tabelele de bază, printr-un trigger *instead of*. Astfel, ELEARN_APP_ADMIN doreşte să creeze vizualizarea VIZ_FEEDB:
			`CREATE OR REPLACE VIEW VIZ_FEEDB AS`
			`SELECT MESAJ,COD_STUDENT,'PROF1' AS cod_prof`
			`FROM ELEARN_profesor1.FEEDBACK`
			`UNION`
			`SELECT MESAJ,COD_STUDENT,'PROF2' AS cod_prof`
			`FROM ELEARN_profesor2.FEEDBACK`
			`UNION`
			`SELECT MESAJ,COD_STUDENT,'ASIST3' AS cod_prof`
			`FROM ELEARN_asistent3.FEEDBACK;`
			• Inițial ELEARN_APP_ADMIN va primi eroare de privilegii insuficiente.

			• Pentru a rezolva această problema, SYS/AS SYSDBA îi acordă următoarele privilegii:
			○ **privilegiu sistem de creare de view in schema proprie (1.1):**
			`GRANT CREATE VIEW TO ELEARN_APP_ADMIN ;`
			==> Nu este suficient, continua sa primeasca eroare de privilegii insuficiente
			○ **privilegiu obiect de select pe fiecare din tabelele ce intervin in cererea din vizualizare, with grant option:**
			`GRANT SELECT ON ELEARN_profesor1.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			`GRANT SELECT ON ELEARN_profesor2.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			`GRANT SELECT ON ELEARN_asistent3.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			==> acum vizualizarea este creată cu succes de către ELEARN_APP_ADMIN .
			• Îi dăm unui student privilegiul de a interoga vizualizarea:
			`GRANT SELECT ON ELEARN_APP_ADMIN.VIZ_FEEDB TO ELEARN_student1;`
			• Studentul interogheaza vizualizarea cu succes:
			`SELECT * FROM ELEARN_APP_ADMIN.VIZ_FEEDB WHERE cod_student=1;`
			• În ceea ce priveşte inserarea de date în tabelele de bază prin intermediul vizualizării: întrucât vizualizarea conține operatorul UNION ⟹ vizualizarea este complexă ⟹ nu se pot insera date direct prin *view*, ci este necesar un *trigger* INSTEAD OF.
			• Este necesar ca utilizatorul ELEARN_APP_ADMIN să poată crea triggeri în schema proprie. SYS AS SYSDBA îi acordă privilegiul:
			`GRANT CREATE TRIGGER TO ELEARN_APP_ADMIN;`
			• În plus, ELEARN_APP_ADMIN trebuie să primească privilegii de a insera în tabelele de bază ale vizualizării, cu clauza *grant option*:
			`GRANT INSERT ON ELEARN_profesor1.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			`GRANT INSERT ON ELEARN_profesor2.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			`GRANT INSERT ON ELEARN_asistent3.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			• ELEARN_APP_ADMIN creeaza un trigger INSTEAD OF astfel:

```
CREATE OR REPLACE TRIGGER TR_FEEDB
INSTEAD OF INSERT ON VIZ_FEEDB
FOR EACH ROW
BEGIN
   IF :NEW.COD_PROF='PROF1' THEN
      INSERT INTO ELEARN_profesor1.FEEDBACK
      VALUES(SYSDATE- TO_DATE('2000-01-01', 'yyyy-mm-dd'),
             :NEW.MESAJ,101,SYSDATE);
   END IF;
   IF :NEW.COD_PROF='PROF2' THEN
      INSERT INTO ELEARN_profesor2.FEEDBACK
      VALUES(SYSDATE- TO_DATE('2000-01-01', 'yyyy-mm-dd'),
             :NEW.MESAJ,101,SYSDATE);
   END IF;
   IF :NEW.COD_PROF='ASIST3' THEN
      INSERT INTO ELEARN_asistent3.FEEDBACK
      VALUES(SYSDATE- TO_DATE('2000-01-01', 'yyyy-mm-dd'),
             :NEW.MESAJ,101,SYSDATE); END IF;
END;
/
Irigger created.
```

- Acum SYS (AS SYSDBA) acordă privilegiul de insert prin *view* către studentul ELEARN_student1:

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

- Studentul executa următorul insert:

```
INSERT INTO ELEARN_APP_ADMIN.VIZ_FEEDB VALUES('UN CURS
INTERESANT',101,'PROF1');
SQL> INSERT INTO ELEARN_APP_ADMIN.VIZ_FEEDB VALUES

1 row created.

COMMIT; -- neaparat !
```

- Profesorul ELEARN_profesor1 verifică dacă a primit vreun *feedback*:

```
SELECT ID, SUBSTR(MESAJ,1,20) MESAJ, COD_STUDENT,TIMP 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) Privilegii procedură
		
		EXECUTE ON schema.proc
			Există două situații:
			• **Situația 1)**, similar *view*-ului și *trigger*-ului create anterior, creatorul procedurii primește drept de a crea o procedură în schema proprie și, în plus, el primește privilegii corespunzătoare pe obiectele prelucrate în interiorul procedurii, cu clauza WITH GRANT OPTION.
			• În acest caz, apelantul are nevoie doar de privilegiul *execute* asupra procedurii.
			• Astfel, ELEARN_APP_ADMIN doreşte sa creeze procedura DELETE_SPAM care sa stearga comentariile de tip SPAM din tabela de FEEDBACK a profesorului. Ca input procedura primeşte minim 1, maxim 3 cuvinte cheie de SPAM ce vor fi cautate in mesaje:
			`CREATE OR REPLACE PROCEDURE DELETE_SPAM(key1 VARCHAR2, key2 VARCHAR2 default 'reclama', key3 VARCHAR2 default 'publicitate') AS`
			`BEGIN`
			`DELETE FROM ELEARN_profesor1.FEEDBACK`
			`WHERE MESAJ LIKE '%'||key1||'%' OR MESAJ LIKE '%'||key2||'%'`
			`OR MESAJ LIKE '%'||key3||'%';`
			`DBMS_OUTPUT.PUT_LINE('Au fost sterse:'||SQL%ROWCOUNT||' mesaje de tip spam din tabela profesorului 1');`
			`COMMIT;`
			`END;`
			`/`
			`CREATE OR REPLACE PROCEDURE DELETE_SPAM(`
			`licitate') DELETE_SPAM AS`
			`*`
			`ERROR at line 1:`
			`ORA-01031: insufficient privileges`
			• Pentru a rezolva această problemă, SYS/AS SYSDBA îi acordă următoarele privilegii lui ELEARN_APP_ADMIN:
			* privilegiul de a crea proceduri în schema proprie:
			`GRANT CREATE PROCEDURE TO ELEARN_APP_ADMIN;`
			* privilegii asupra obiectelor prelucrate de procedură, cu GRANT OPTION:
			`GRANT DELETE ON ELEARN_profesor1.FEEDBACK TO ELEARN_APP_ADMIN WITH GRANT OPTION;`
			• Acum , ELEARN_APP_ADMIN va reuşi să creeze procedura cu succes.
			• Mai mult, utilizatorul ELEARN_APP_ADMIN poate executa procedura cu succes:

```
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.
```

- Presupunem că proiectantul aplicației a stabilit că acest task de ștergere a mesajelor de *spam* să poată fi realizat și de către un asistent. Când acesta încearcă să execute, primește eroare:

```
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
```

- Prin urmare, asistentul ELEARN_asistent3 primește de la SYS (AS SYSDBA) următoarele privilegii:

```
GRANT EXECUTE ON ELEARN_APP_ADMIN.DELETE_SPAM TO
ELEARN_asistent3;
```

- Acum, și asistentul va putea executa cu succes procedura.

```
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.
```

- **Situația 2)** în cazul în care creatorul procedurii are privilegii corespunzătoare pe obiectele prelucrate în interiorul procedurii, dar fara *grant option*, ATUNCI apelantul va avea nevoie să aibă el însuși drepturile respective, altfel neputând executa cu succes procedura.

4. Recapitularea situaţiilor întâlnite in exemple

userul X creeaza un obiect de tip view ⟨trigger, procedura⟩

	in schema proprie			*in schema altui user Y*		
	acceseaza obiecte din schema proprie	*acceseaza obiecte din schema lui Y* **(select Y.D, insert Y.D)**		*acceseaza obiecte din schema proprie*	*acceseaza obiecte din schema lui Y* **(select Y.D, insert Y.D)**	
Ce privilegii are nevoie X	*CREATE VIEW*	*CREATE VIEW*		*CREATE ANY VIEW*	*CREATE ANY VIEW*	
		SELECT ON Y.D *INSERT ON Y.D*	*SELECT ON Y.D WITH GRANT OPTION* *INSERT ON Y.D WITH GRANT OPTION*		*SELECT ON Y.D* *INSERT ON Y.D*	*SELECT ON Y.D WITH GRANT OPTION* *INSERT ON Y.D WITH GRANT OPTION*
Ce privilegii are nevoie apelantul Z	*SELECT ON VIZ* *INSERT ON VIZ*	*SELECT ON VIZ* *INSERT ON VIZ* *SELECT ON Y.D* *INSERT ON Y.D*	*SELECT ON VIZ* *INSERT ON VIZ*	*SELECT ON VIZ* *INSERT ON VIZ*	*SELECT ON VIZ* *INSERT ON VIZ* *SELECT ON Y.D* *INSERT ON Y.D*	*SELECT ON VIZ* *INSERT ON VIZ*

5. Roluri

- **Rolurile** sunt containere pentru privilegii, astfel încât sa permită o mai uşoară administrare a acestora: atunci când un utilizator primeşte un rol, implicit primeşte toate privilegiile conţinute în rolul respectiv.

- Există roluri predefinite în Oracle, de exemplu:

Rolul	**Privilegiile conținute în rol**
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	Include toate privilegiile, cu opțiune de administrare a lor (opțiunea de a fi acordate mai departe)

Observație: SYSDBA este un caz special de rol, asemănător DBA
Retinem! A NU se confunda SYS , care este un utilizator, cu SYSDBA, care este un rol.

- **Sintaxa *:***
  - Crearea unui rol:
    ```
    CREATE ROLE numerol;
    ```
  - Atribuirea unui rol catre un utilizator :
    ```
    GRANT numerol TO utilizator [WITH ADMIN OPTION];
    ```
  - Includerea unor noi privilegii in rolul creat. Acestea vor fi preluate implicit de utilizatorii rolului(daca nu exista contradictii – vezi ierarhia rolurilor in capitolul urmator):
    ```
    GRANT privilegiu_1,privilegiu_2,...,privilegiu_n [ON obiect]
    TO numerol;
    ```

- Aflarea rolurilor utilizatorilor aplicaţiei de e-learning se poate realiza cu comanda:

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

- Utilizarea rolurilor prezintă avantajul unui management mai ușor al privilegiilor, însă prezintă și anumite dezavantaje:
  - În cadrul procedurilor rolurile sunt inhibate, nu au efect . Astfel, privilegiul necesar va trebui acordat individual și direct utilizatorului, nu prin rol;
  - Câte roluri poate avea simultan un utilizator? Răspuns: zero, unul sau mai multe.

Exemplu:

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

CREATE ROLE update_tot;
GRANT UPDATE ANY TABLE TO update_tot;


GRANT select_tot TO
ELEARN_APP_ADMIN; GRANT
update_tot 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. Ierarhia priorităților de roluri și privilegii

- Există reguli privind agregarea și prioritizarea privilegiilor unui utilizator.
- Privilegiile și rolurile pot fi văzute ca modalități de a da anumite drepturi, dar și de a impune anumite restricții. Acest lucru se realizează prin mecanismul GRANT și REVOKE pentru privilegii și roluri.

Recapitulăm: ELEARN_APP_ADMIN, ca proprietar al tabelului REZOLVA, execută comenzile din tabelul următor:

UTILIZATORUL ELEARN_student1 **NU ARE PRIVILEGII PE TABELA REZOLVA**				
PRIVILEGIU SELECT DAT USERULUI DIRECT	**PRIVILEGIU SELECT DAT ROLULUI**	**PRIVILEGIU SELECT DAT ROLULUI**	**PRIVILEGIU SELECT DAT ROLULUI**	**PRIVILEGIU SELECT DAT USERULUI DIRECT**
GRANT SELECT ON REZOLVA TO ELEARN_student1;	CREATE ROLE rol_stud; GRANT SELECT ON REZOLVA TO rol_stud;	CREATE ROLE rol_stud; GRANT SELECT ON REZOLVA TO rol_stud;	CREATE ROLE rol_stud; GRANT SELECT ON REZOLVA TO rol_stud;	GRANT SELECT ON REZOLVA TO ELEARN_student1;
	ACORD ROLUL UTILIZATORULUI	**ACORD ROLUL UTILIZATORULUI**	**ACORD ROLUL UTILIZATORULUI**	**PRIVILEGIU SELECT DAT ROLULUI**
	GRANT rol_stud TO ELEARN_student1;	GRANT rol_stud TO ELEARN_student1;	GRANT rol_stud TO ELEARN_student1;	CREATE ROLE rol_stud; GRANT SELECT ON REZOLVA TO rol_stud;
		PRIVILEGIU SELECT REVOCAT USERULUI DIRECT	**PRIVILEGIU SELECT REVOCAT ROLULUI**	**ACORD ROLUL UTILIZATORULUI**
		REVOKE SELECT ON REZOLVA FROM ELEARN_student1;	REVOKE SELECT ON REZOLVA FROM rol_stud;	GRANT rol_stud TO ELEARN_student1;
				PRIVILEGIU SELECT REVOCAT ROLULUI
				REVOKE SELECT ON REZOLVA FROM rol_stud;
SUCCES	**SUCCES**	**Eroare**	**ESEC**	**SUCCES !**

Tabelul 2

Observaţii:

- Un privilegiu acordat în mod direct utilizatorului rămâne valabil chiar dacă un rol al lui care anterior îl cuprindea îl pierde.
- De asemenea, proprietarul unui obiect are toate privilegiile asupra lui, cu ADMIN option. Nimeni nu ii poate revoca vreodată vreun privilegiu pe un obiect din schema proprie.
- Granularitatea de acordare (GRANT) a privilegiilor trebuie respectată în oglindă la retragere (REVOKE):

```
GRANT CREATE ANY TABLE TO ELEARN_asistent3;
    ⟶ REVOKE CREATE ANY TABLE FROM ELEARN_asistent3; -- corect
    ⟶ REVOKE CREATE TABLE FROM ELEARN_asistent3; -- incorect
```

- Dacă un utilizator primește un privilegiu doar printr-un rol, nu direct, atunci privilegiul respectiv nu i se poate retrage direct cu *revoke*.
- Reţinem că REVOKE poate fi dat numai la nivelul întregului tabel, nu la nivel de coloane individuale.

Exemplu:

```
GRANT UPDATE(deadline) ON ELEARN_APP_ADMIN.TEMA_CASA TO
ELEARN_asistent3;
REVOKE UPDATE ON ELEARN_APP_ADMIN.TEMA_CASA FROM ELEARN_asistent3;
```

7. Exerciții

Construirea matricii entitate –utilizator , rezultată din matricile proces-utilizator și entitate-proces.

	Studenţii cu frecventa	Studenţii la distanta	Profesorii	Asistenţii	Secretarii	Alumnii	Admin aplicatie si BD	Public larg
FISA_DISCIPLINA	S	S	S	S	S	S	S	S
CURS	S	S	I,U,S	S	S	S	I,U,S	S
MATERIAL_STUDIU	S	S	I,U,D,S	I,U,D,S				
TEMA_CASA	S	S	I,U,S	I,U,S				
NOTA	S	S	S	S	S			
UTILIZATOR							I,U	
CURSANT	S	S	S	S	S		S,I,U	
CADRU_DIDACTIC	S	S					I,U	
EVALUARE	S	S	I,U,S		I,U,S			
PARTICIPA	S	S	I,U	I,U	S		I,U,S	
PREDA	S	S	I,U				I,U	
REZOLVA	I,U	I,U	U	U				
SUSTINE	S	S	I,U		S			
FEEDBACK	I,U,D	I,U,D	S	S				

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

1. **Utilizați trei modalități diferite de a da drept celor doi utilizatori profesori să obțină informații despre coloanele tabelului *TEMA_CASA*.**

Indicație:

- privilegii asupra obiectelor schemei acordate pe tabel direct către utilizatori;
- privilegii vizualizare acordate direct către utilizatori, *view*-ul fiind în schema admin-ului; rol care include privilegiile asupra obiectelor schemei.

2. **Utilizați trei modalități diferite de a da drept utilizatorilor cadre didactice să actualizeze prin aplicație *deadline*-ul temelor de casă (coloanele tabelului *TEMA_CASA*), fără a putea modifica restul informațiilor din temă.**

Indicație:

- privilegii asupra obiectelor schemei acordate pe tabel direct către utilizatori;
- privilegii vizualizare acordate direct către utilizatori, *view*-ul fiind in schema admin-ului;
- rol care include privilegiile asupra obiectelor schemei.

3. **Creați o procedură *PROC_NOTARE* care să permită notarea temelor de casă. Procedura va fi în schema admin-ului, dar va putea fi apelată de profesori și asistent. Procedura va primi ca parametri de intrare codul studentului, codul temei, codul cadrului didactic corector și nota acordata. În *background*, procedura va face prelucrări care să verifice că tema aparține studentului indicat și că nu este deja notată.**

4. Creați un context de privilegii la nivelul utilizatorului de tip student care să fie repetabil pentru orice student din sistem. Contextul va face diferențiere pentru studenții din anul 3 (terminal licență) și pentru studenții din anul 5 (terminal master) care nu mai trimit teme de casă (exemplu strict educațional).