Securitatea bazelor de date

*Gestionarea pieselor de teatru*

1. Introducere

1.1 Prezentarea succintă a modelului proiectat și a regulilor sale

Baza de date aleasă poate fi folosită în cadrul unui *sistem de vânzare de bilete de teatru*. Aceasta va conține diverse entități, precum piesele de teatru care pot fi vizionate, reprezentațiile acestora, actorii participanți, locațiile în care au loc, precum și sălile disponibile. Un spectator își poate achiziționa un bilet la o reprezentație, împreună cu un număr de locuri, dacă mai sunt disponibile.  
 Printre utilizatorii bazei de date, putem menționa spectatorii, care vor putea achiziționa bilete la o piesă de teatru și vor putea lăsa recenzii, actorii, care vor răspunde la recenziile spectatorilor, dar și consultanții artistici, care sunt responsabili cu adăugarea pieselor de teatru, a reprezentațiilor, cât și a locațiilor în sistem.

* 1. Diagrama conceptuală

Identificarea entităților prin modelarea datelor aplicației:

* Baza de date reprezintă un sistem de gestiune al biletelor pieselor de teatru
* În sistem pot fi adăugate piesele de teatru, reprezentațiile, respectiv teatrele și sălile în care acestea au loc. De asemenea, fiecărei piese de teatru îi sunt asociați actorii.
* La o reprezentație își pot cumpăra unul sau mai multe bilete unul sau mai mulți spectatori. Un spectator își poate vizualiza, șterge și modifica datele numai de pe *propriile bilete*. Un spectator poate lăsa recenzii pieselor de teatru. De asemenea, poate șterge și modifica datele numai de pe *propriile recenzii.*
* Un actor poate răspunde recenziilor date de spectatori. Totodată, poate modifica sau șterge numai *propriile răspunsuri.* Toți utilizatorii pot vedea numai *anumite date din profilul actorilor*, însă un actor își poate *vedea toate datele proprii* și le poate modifica.

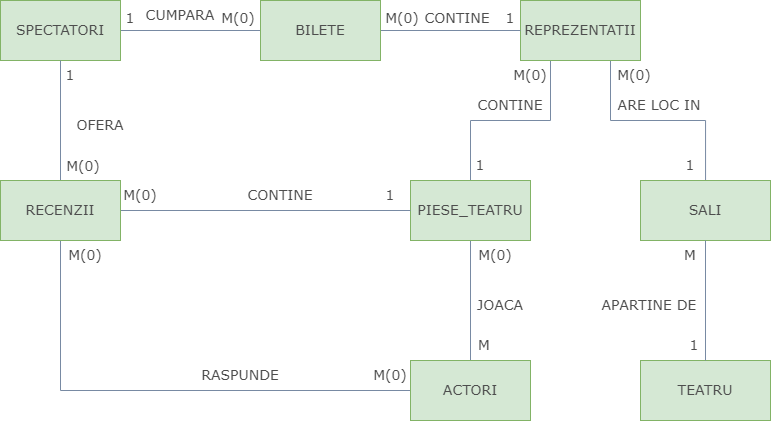


Diagrama E/R

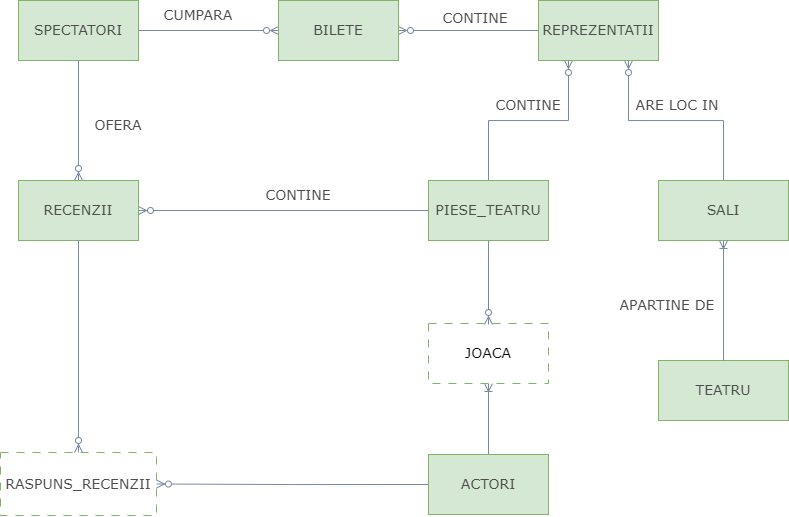


Diagrama Conceptuală

* 1. Schemele relaționale

**SPECTATORI** (spectator\_id#, username, parola, prenume, nume, email, numar\_telefon)

**ACTORI** (actor\_id#, username, parola, prenume, nume, email, numar\_telefon, locul\_nasterii, data\_nasterii, biografie)

**JOACA**(piesa\_id, actor\_id)

**PIESE\_TEATRU**(piesa\_id#, titlu, autor, regizor, durata, gen)

**TEATRU**(teatru\_id#, nume, oras)

**SALI**(sala\_id#, nume, capacitate, teatru\_id)

**REPREZENTATII** (reprezentatie\_id#, piesa\_id, sala\_id, data\_reprezentatie, pret, ora, status)

**BILETE** (reprezentatie\_id, spectator\_id, nr\_locuri)

**RECENZII** (recenzie\_id#, spectator\_id, piesa\_id, mesaj, nota, data\_recenzie)

**RASPUNS\_RECENZII** (raspuns\_id#, recenzie\_id, actor\_id, mesaj, data\_recenzie)

1.3. Crearea tabelelor

Metodele de criptare din insert vor fi menționate în capitolul următor.

SPECTATORI

CREATE TABLE spectatori

(spectator\_id NUMBER(5) PRIMARY KEY,

username VARCHAR2(20) NOT NULL,

parola RAW(100) NOT NULL,

prenume VARCHAR2(20) NOT NULL,

nume VARCHAR2(20) NOT NULL,

email RAW(100) NOT NULL UNIQUE,

numar\_telefon RAW(100) UNIQUE);

CREATE SEQUENCE spectatori\_seq

START WITH 1

INCREMENT BY 1;

CREATE OR REPLACE PROCEDURE insert\_spectatori(username VARCHAR2, parola VARCHAR2, prenume VARCHAR2, nume VARCHAR2, email VARCHAR2, numar\_telefon VARCHAR2) AS

id\_ NUMBER(5);

parola\_hash RAW(100);

email\_cript RAW(100);

telefon\_cript RAW(100);

BEGIN

id\_ := spectatori\_seq.NEXTVAL;

parola\_hash := hash\_parola(parola);

email\_cript := criptare\_aes(email, id\_, 'spectatori');

telefon\_cript := criptare\_aes(numar\_telefon, id\_, 'spectatori');

INSERT INTO spectatori(spectator\_id, username, parola, prenume, nume, email, numar\_telefon)

VALUES (id\_, username, parola\_hash, prenume, nume, email\_cript, telefon\_cript);

COMMIT;

END;

/

EXECUTE insert\_spectatori('ipoop', 'ipoop', 'Ioana', 'Pop', 'ioanapop@gmail.com', '0721174482');

EXECUTE insert\_spectatori('dediu', 'dediu', 'Marius', 'Dediu', 'marius@gmail.com', '0787123832');

EXECUTE insert\_spectatori('avramescu', 'avramescu', 'Victor', 'Avram', 'avram\_vic@gmail.com', '0761124428');

EXECUTE insert\_spectatori('epitaru', 'epitaru', 'Erika', 'Pitaru', 'pit\_erika@gmail.com', '0719921004');

EXECUTE insert\_spectatori('alexpopa', 'alexpopa', 'Alexandra', 'Popa', 'alex\_pop@gmail.com', '0721193392');

CREATE SEQUENCE spectatori\_seq

START WITH 1

INCREMENT BY 1;

CREATE OR REPLACE PROCEDURE insert\_spectatori(username VARCHAR2, parola VARCHAR2, prenume VARCHAR2, nume VARCHAR2, email VARCHAR2, numar\_telefon VARCHAR2) AS

id\_ NUMBER(5);

parola\_hash RAW(100);

email\_cript RAW(100);

telefon\_cript RAW(100);

BEGIN

id\_ := spectatori\_seq.NEXTVAL;

parola\_hash := hash\_parola(parola);

email\_cript := criptare\_aes(email, id\_, 'spectatori');

telefon\_cript := criptare\_aes(numar\_telefon, id\_, 'spectatori');

INSERT INTO spectatori(spectator\_id, username, parola, prenume, nume, email, numar\_telefon)

VALUES (id\_, username, parola\_hash, prenume, nume, email\_cript, telefon\_cript);

COMMIT;

END;

/

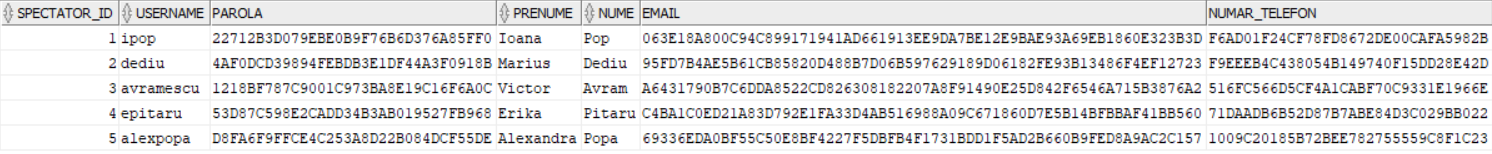
EXECUTE insert\_spectatori('ipoop', 'ipoop', 'Ioana', 'Pop', 'ioanapop@gmail.com', '0721174482');

EXECUTE insert\_spectatori('dediu', 'dediu', 'Marius', 'Dediu', 'marius@gmail.com', '0787123832');

EXECUTE insert\_spectatori('avramescu', 'avramescu', 'Victor', 'Avram', 'avram\_vic@gmail.com', '0761124428');

EXECUTE insert\_spectatori('epitaru', 'epitaru', 'Erika', 'Pitaru', 'pit\_erika@gmail.com', '0719921004');

EXECUTE insert\_spectatori('alexpopa', 'alexpopa', 'Alexandra', 'Popa', 'alex\_pop@gmail.com', '0721193392');



ACTORI

CREATE TABLE actori

(actor\_id NUMBER(5) PRIMARY KEY,

username VARCHAR2(20) NOT NULL,

parola RAW(100) NOT NULL,

prenume VARCHAR2(20) NOT NULL,

nume VARCHAR2(20) NOT NULL,

email RAW(100) NOT NULL UNIQUE,

numar\_telefon RAW(100) NOT NULL UNIQUE,

locul\_nasterii VARCHAR2(20),

data\_nasterii DATE,

biografie VARCHAR2(500));

CREATE SEQUENCE actori\_seq

START WITH 1

INCREMENT BY 1;

CREATE OR REPLACE PROCEDURE insert\_actori(username VARCHAR2, parola VARCHAR2, prenume VARCHAR2, nume VARCHAR2, email VARCHAR2, numar\_telefon VARCHAR2, locul\_nasterii VARCHAR2, data\_nasterii DATE, biografie VARCHAR2) AS

id\_ NUMBER(5);

parola\_hash RAW(100);

email\_cript RAW(100);

telefon\_cript RAW(100);

BEGIN

id\_ := actori\_seq.NEXTVAL;

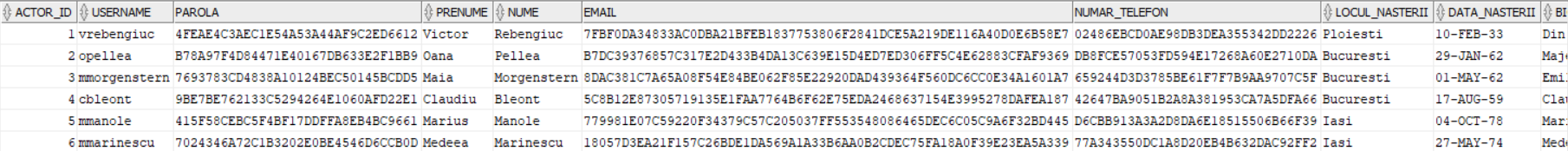
parola\_hash := hash\_parola(parola);

email\_cript := criptare\_aes(email, id\_, 'actori');

telefon\_cript := criptare\_aes(numar\_telefon, id\_, 'actori');

INSERT INTO actori (actor\_id, username, parola, prenume, nume, email, numar\_telefon, locul\_nasterii, data\_nasterii, biografie)

VALUES (id\_, username, parola\_hash, prenume, nume, email\_cript, telefon\_cript, locul\_nasterii, data\_nasterii, biografie);



COMMIT;

END;

/

EXECUTE insert\_actori('vrebengiuc', 'vrebengiuc', 'Victor', 'Rebengiuc', 'rebengiuc@gmail.com', '0728119013', 'Bucuresti', TO\_DATE('10/02/1933', 'DD/MM/YYYY'), 'Din 1957, a ….');

EXECUTE insert\_actori('opellea', 'opellea', 'Oana', 'Pellea', 'pellea@gmail.com', '0728814491', 'Bucuresti', TO\_DATE('29/01/1962', 'DD/MM/YYYY'), 'Majestatea Sa Regele Mihai I ….');

EXECUTE insert\_actori('mmorgenstern', 'mmorgenstern', 'Maia', 'Morgenstern', 'morgenstern@yahoo.com', '0734491191','Bucuresti', TO\_DATE('01/05/1962', 'DD/MM/YYYY'), 'Emilia-Maia-Ninel Morgenstern s-a …. ');

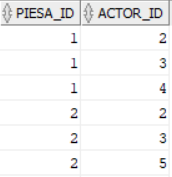
EXECUTE insert\_actori('cbleont', 'cbleont', 'Claudiu', 'Bleont', 'bleont@yahoo.com', '0718112749', 'Bucuresti', TO\_DATE('17/08/1959', 'DD/MM/YYYY'), 'Claudiu Bleont…');

EXECUTE insert\_actori('mmanole', 'mmanole', 'Marius', 'Manole', 'manole@gmail.com', '0745518172', 'Iasi', TO\_DATE('04/10/1978', 'DD/MM/YYYY'), 'Marius Manole …');

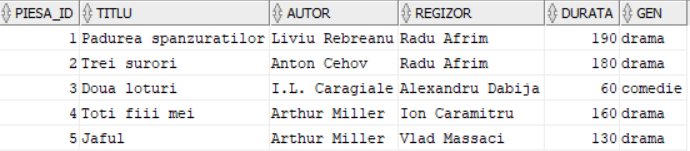
EXECUTE insert\_actori('mmarinescu', 'mmarinescu', 'Medeea', 'Marinescu', 'marin11@gmail.com', '0718442812', 'Iasi', TO\_DATE('27/05/1974', 'DD/MM/YYYY'), 'Medeea Marinescu …');

În continuare vom prezenta numai conținutul tabelelor, întrucât codul pentru creare și inserare poate fi găsit separat.

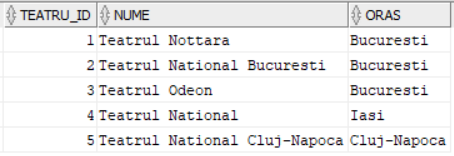
JOACA



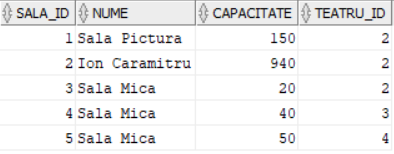
PIESE\_TEATRU



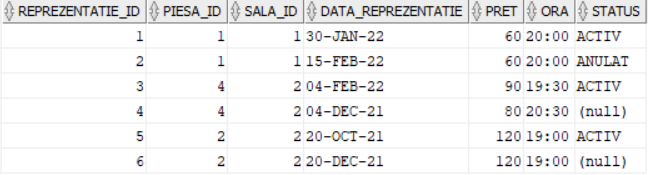
TEATRU



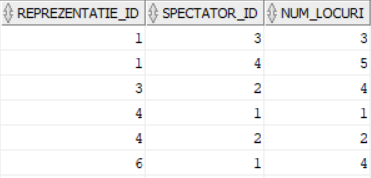
SALI

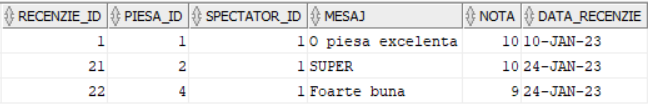


REPREZENTATII

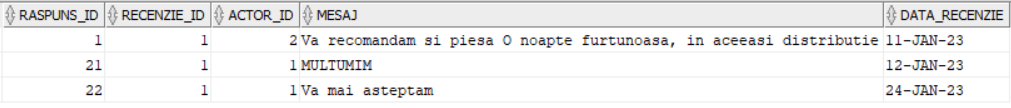


BILETE

  
RECENZII



RASPUNS\_RECENZII



1.4. Prezentarea regulilor de securitate aplicate asupra modelului

În ceea ce privește regulilte de securitate care vor fi aplicate asupra modelului, putem menționa:

* Criptarea coloanelor *email* și *numar\_telefon* corespunzătoare tabelelor SPECTATORI și ACTORI, folosind **AES**
* Aplicarea algoritmului de *hashing* **MD5** asupra parolelor spectatorilor și actorilor
* Auditarea logării utilizatorilor, a diverselor operațiuni asupra bazei de date, odată cu crearea trigger-ilor și politicilor de auditare
* Definirea utilizatorilor: consultanți artistici, spectatori, actori, admin, utilizatorul la nivel de SO și guest
* Fiecare utilizator va avea un *quota* asociat, un plan de resurse configurat și profiluri de utilizatori care determină reguli asupra parolelor
* Atribuirea privilegiilor diferite asupra tabelelor pentru fiecare tip de utilizator, definind și *vizualizări* în schema adminului pentru ca anumite operații să nu poată fi făcute decât pe secțiuni din tabele, fără a avea acces la tot conținutul
* Adăugarea în contextul aplicației a regulilor cu privire la intervalul orar în care se pot efectua ștergeri sau modificări pe REPREZENTATII
* Testarea posibilelor atacuri de securitate prin SQL Injection
* Mascarea datelor din tabelele ACTORI și SPECTATORI

1. Criptarea datelor

În cazul criptării datelor, atât pentru tabelul ACTORI, cât și pentru tabelul SPECTATORI, se vor cripta coloanele *email* și *numar\_telefon*. Algoritmul de criptare folosit este **AES** (*Advanced Encryption Standard*), întrucât permite alegerea unor chei de 128, 192 sau 256 de biți, fiind exponențial mai puternic față de DES, care permite alegerea unei chei de numai 56 de biți. De asemenea, AES este mult mai rapid față de DES și în special față de 3DES (fiind cel mai lent dintre cele 3). Nu în ultimul rând, AES este considerat sigur în fața atacurilor de tip brute-force, spre deosebire de DES și 3DES, care au putut fi sparte cu astfel de tipuri de atac.

Pentru criptare am folosit funcțiile din pachetul *DBMS\_CRYPTO*:

* *encrypt\_aes128* pentru criptarea AES cu o cheie de 128 de biți
* *pad\_pkcs5* în cazul padding-ului
* *chain\_cbc* (Cipher Block Chaining) în cazul tehnicii de chaining, unde pentru fiecare bloc se va face operația XOR cu un vector înaintea criptării

În cazul cheilor de criptare, am utilizat o cheie la nivelul tabelului și câte o cheie la nivel de înregistrare, în final fiind obținută o cheie hibridă, în urma operației XOR aplicată între cheile rezultate anterior. Astfel, dacă s-ar divulga cheia master și o cheie de înregistrare, celelalte înregistrări ar fi în continuare în siguranță. Am creat două tabele (*CHEI\_TABEL* și *CHEI\_INREGISTRARE*), pentru a stoca cheile la nivelul tabelelor, respectiv la nivel de înregistrare.

CREATE TABLE chei\_inregistrare(

id\_cheie NUMBER(5) PRIMARY KEY,

cheie raw(16) NOT NULL,

nume\_tabel VARCHAR2(30) NOT NULL,

id\_inregistrare NUMBER(5) NOT NULL);

CREATE SEQUENCE chei\_inregistrare\_seq

START WITH 1

INCREMENT BY 1;

CREATE TABLE chei\_tabel(

id\_cheie NUMBER(5) PRIMARY KEY,

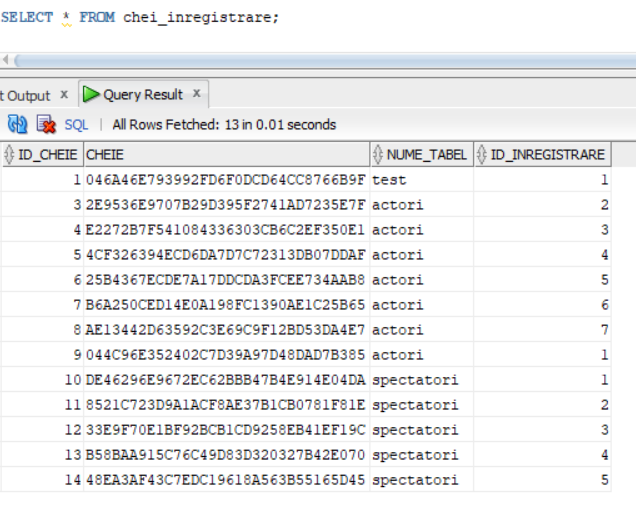
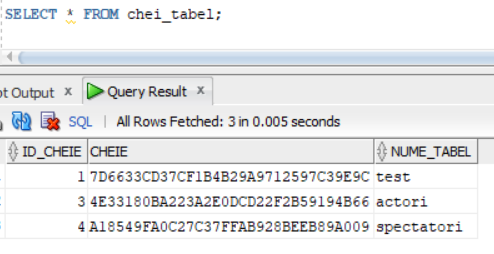
cheie raw(16) NOT NULL,

nume\_tabel VARCHAR2(30) NOT NULL);

CREATE SEQUENCE chei\_tabel\_seq

START WITH 1

INCREMENT BY 1;



În vederea obținerii criptării, am creat funcțiile *criptare\_aes* (respectiv *decriptare\_aes* pentru decriptare). Pentru a demonstra eficiența criptării, am creat metoda *test\_decriptare* care va decripta email-ul primei înregistrări din tabelul actori.

CREATE OR REPLACE FUNCTION criptare\_aes(valoare VARCHAR2, inregistrare NUMBER, tabel VARCHAR2) RETURN RAW AS

cheie\_tabel RAW(16);

exista\_cheie\_tabel NUMBER(1);

cheie\_inregistrare RAW(16);

exista\_cheie\_inregistrare NUMBER(1);

mod\_operare PLS\_INTEGER;

raw\_valoare RAW(100);

cheie\_xor RAW(100);

valoare\_cript RAW(100);

BEGIN

SELECT COUNT(\*) INTO exista\_cheie\_tabel

FROM chei\_tabel

WHERE nume\_tabel = tabel;

IF exista\_cheie\_tabel = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista cheie asociata la nivel de tabel');

cheie\_tabel := dbms\_crypto.randombytes(16);

INSERT INTO chei\_tabel VALUES (chei\_tabel\_seq.NEXTVAL, cheie\_tabel, tabel);

ELSE

SELECT cheie INTO cheie\_tabel

FROM chei\_tabel

WHERE nume\_tabel = tabel;

END IF;

SELECT COUNT(\*) INTO exista\_cheie\_inregistrare

FROM chei\_inregistrare

WHERE nume\_tabel = tabel

AND id\_inregistrare = inregistrare;

IF exista\_cheie\_inregistrare = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista cheie asociata la nivel de inregistrare');

cheie\_inregistrare := dbms\_crypto.randombytes(16);

DBMS\_OUTPUT.PUT\_LINE('Cheie inregistrare: ' || cheie\_inregistrare);

INSERT INTO chei\_inregistrare VALUES (chei\_inregistrare\_seq.NEXTVAL, cheie\_inregistrare, tabel, inregistrare);

ELSE

SELECT cheie INTO cheie\_inregistrare

FROM chei\_inregistrare

WHERE nume\_tabel = tabel

AND id\_inregistrare = inregistrare;

END IF;

mod\_operare := dbms\_crypto.encrypt\_aes128 + dbms\_crypto.pad\_pkcs5 + dbms\_crypto.chain\_cbc;

raw\_valoare := utl\_i18n.string\_to\_raw(valoare, 'AL32UTF8');

cheie\_xor := UTL\_RAW.bit\_xor(cheie\_tabel, cheie\_inregistrare);

valoare\_cript := dbms\_crypto.encrypt(raw\_valoare, mod\_operare, cheie\_xor);

COMMIT;

RETURN valoare\_cript;

END;

/

Funcția de decriptare:

CREATE OR REPLACE FUNCTION decriptare\_aes(valoare RAW, inregistrare NUMBER, tabel VARCHAR2) RETURN VARCHAR2 AS

cheie\_tabel RAW(16);

cheie\_inregistrare RAW(16);

mod\_operare PLS\_INTEGER;

cheie\_xor RAW(100);

valoare\_decript\_raw RAW(100);

valoare\_decript VARCHAR2(100);

BEGIN

SELECT cheie INTO cheie\_tabel

FROM chei\_tabel

WHERE nume\_tabel = tabel;

SELECT cheie INTO cheie\_inregistrare

FROM chei\_inregistrare

WHERE nume\_tabel = tabel

AND id\_inregistrare = inregistrare;

mod\_operare := dbms\_crypto.encrypt\_aes128 + dbms\_crypto.pad\_pkcs5 + dbms\_crypto.chain\_cbc;

cheie\_xor := UTL\_RAW.bit\_xor(cheie\_tabel, cheie\_inregistrare);

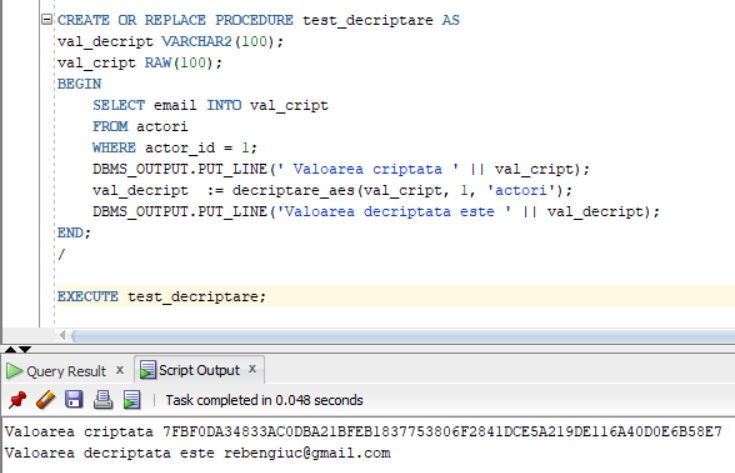
valoare\_decript\_raw := dbms\_crypto.decrypt (valoare, mod\_operare, cheie\_xor);

valoare\_decript := utl\_i18n.raw\_to\_char(valoare\_decript\_raw, 'AL32UTF8');

RETURN valoare\_decript;

END;

/



SELECT cheie INTO cheie\_tabel

FROM chei\_tabel

WHERE nume\_tabel = tabel;

SELECT cheie INTO cheie\_inregistrare

FROM chei\_inregistrare

WHERE nume\_tabel = tabel

AND id\_inregistrare = inregistrare;

mod\_operare := dbms\_crypto.encrypt\_aes128 + dbms\_crypto.pad\_pkcs5 + dbms\_crypto.chain\_cbc;

cheie\_xor := UTL\_RAW.bit\_xor(cheie\_tabel, cheie\_inregistrare);

valoare\_decript\_raw := dbms\_crypto.decrypt (valoare, mod\_operare, cheie\_xor);

valoare\_decript := utl\_i18n.raw\_to\_char(valoare\_decript\_raw, 'AL32UTF8');

RETURN valoare\_decript;

END;

/

În tabelele *ACTORI* și *SPECTATORI* există și coloana *parola*, căreia îi vom aplica algoritmul de hashing MD5, întrucât nu dorim să se poată decripta parola în clar. Am definit funcția *hash\_parola*, care folosește funcția *hash* din pachetul *DBMS\_CRYPTO*.

CREATE OR REPLACE FUNCTION hash\_parola(parola VARCHAR2) RETURN varchar2 AS

md5 raw(16); -- MD5 este 128 = 16 \* 8 bits

BEGIN

md5 := DBMS\_CRYPTO.HASH(

utl\_i18n.string\_to\_raw(parola, 'AL32UTF8'),

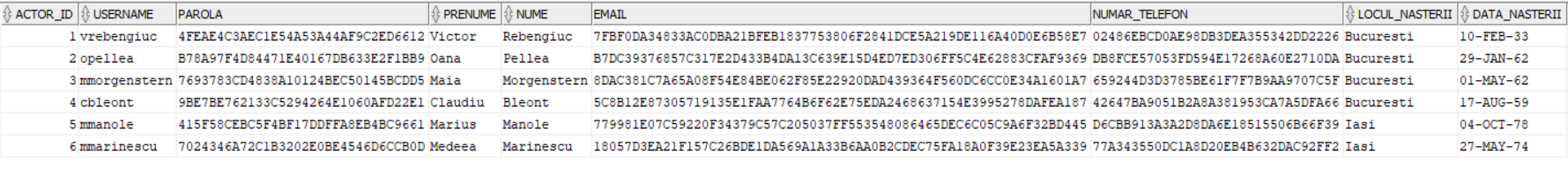
DBMS\_CRYPTO.HASH\_MD5);

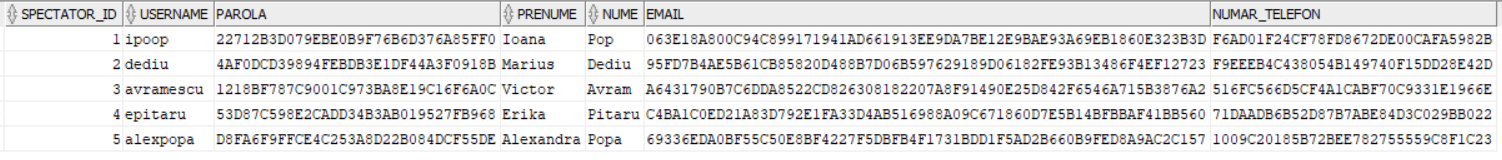
RETURN md5;

END;

/

În final, tabelele *ACTORI* și *SPECTATORI* vor arăta astfel, populate prin intermediul unor proceduri definite, *insert\_actori* și *insert\_spectatori.*

*ACTORI*

 *SPECTATORI*

1. Auditarea activităților asupra bazei de date
   1. Auditare standard

Vrem să audităm:

* toate operațiile fără succes făcute de consultanții artistici
* crearea oricărei vizualizări
* orice operație de update pe un tabel făcute de utilizatori de tip *actor* sau *spectator*

AUDIT INSERT TABLE BY TEATRU\_CONSULTANT1 WHENEVER NOT SUCCESSFUL;

AUDIT DELETE TABLE BY TEATRU\_CONSULTANT1 WHENEVER NOT SUCCESSFUL;

AUDIT UPDATE TABLE BY TEATRU\_CONSULTANT1 WHENEVER NOT SUCCESSFUL;

AUDIT SELECT TABLE BY TEATRU\_CONSULTANT1 WHENEVER NOT SUCCESSFUL;

AUDIT CREATE ANY VIEW;

AUDIT UPDATE TABLE BY TEATRU\_ACTOR1;

AUDIT UPDATE TABLE BY TEATRU\_ACTOR2;

AUDIT UPDATE TABLE BY TEATRU\_ACTOR3;

AUDIT UPDATE TABLE BY TEATRU\_ACTOR4;

AUDIT UPDATE TABLE BY TEATRU\_ACTOR5;

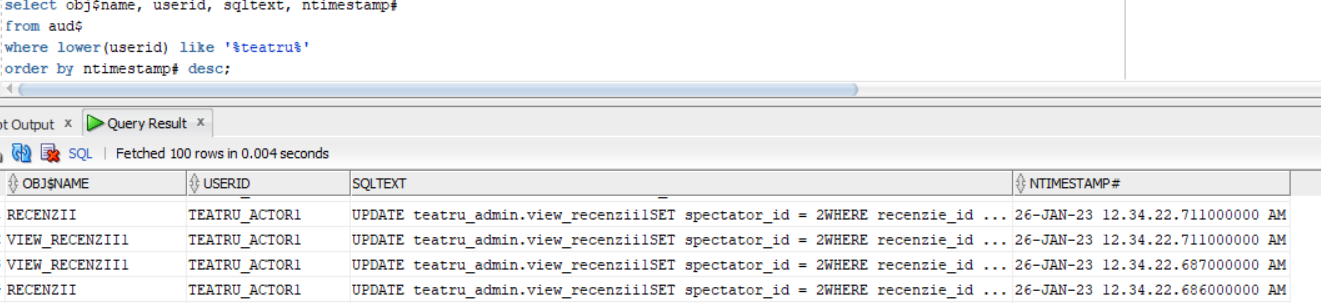
AUDIT UPDATE TABLE BY TEATRU\_SPECTATOR1;

AUDIT UPDATE TABLE BY TEATRU\_SPECTATOR2;

AUDIT UPDATE TABLE BY TEATRU\_SPECTATOR3;

AUDIT UPDATE TABLE BY TEATRU\_SPECTATOR4;

AUDIT UPDATE TABLE BY TEATRU\_SPECTATOR5;



* 1. Triggeri de auditare

Pentru a monitoriza accesul utilizatorilor la baza de date, am creat 2 triggeri, *audit\_conexiune\_logon,* de tip AFTER LOGON ON DATABASE, și *audit\_conexiune\_logoff,* de tip BEFORE LOGOFF ON DATABASE. Aceștia înregistează în tabelul *audit\_conexiune* date despre utilizatorii conectați, împreună cu timpul de login, respectiv logoff.

CREATE OR REPLACE TRIGGER audit\_conexiune\_logon

AFTER LOGON ON DATABASE

BEGIN

IF USER LIKE '%TEATRU%' OR USER LIKE '%LAPTOP%' THEN

INSERT INTO audit\_conexiune VALUES (audit\_conexiune\_seq.nextval, user, sys\_context('userenv', 'sessionid'),

sys\_context('userenv', 'authentication\_method'), sys\_context('userenv', 'authenticated\_identity'),

sys\_context('userenv', 'host'), sysdate, null);

COMMIT;

END IF;

END;

/

CREATE OR REPLACE TRIGGER audit\_conexiune\_logoff

BEFORE LOGOFF ON DATABASE

BEGIN

IF USER LIKE '%TEATRU%' OR USER LIKE '%LAPTOP%' THEN

UPDATE audit\_conexiune

SET timp\_logout = sysdate

WHERE utilizator = user

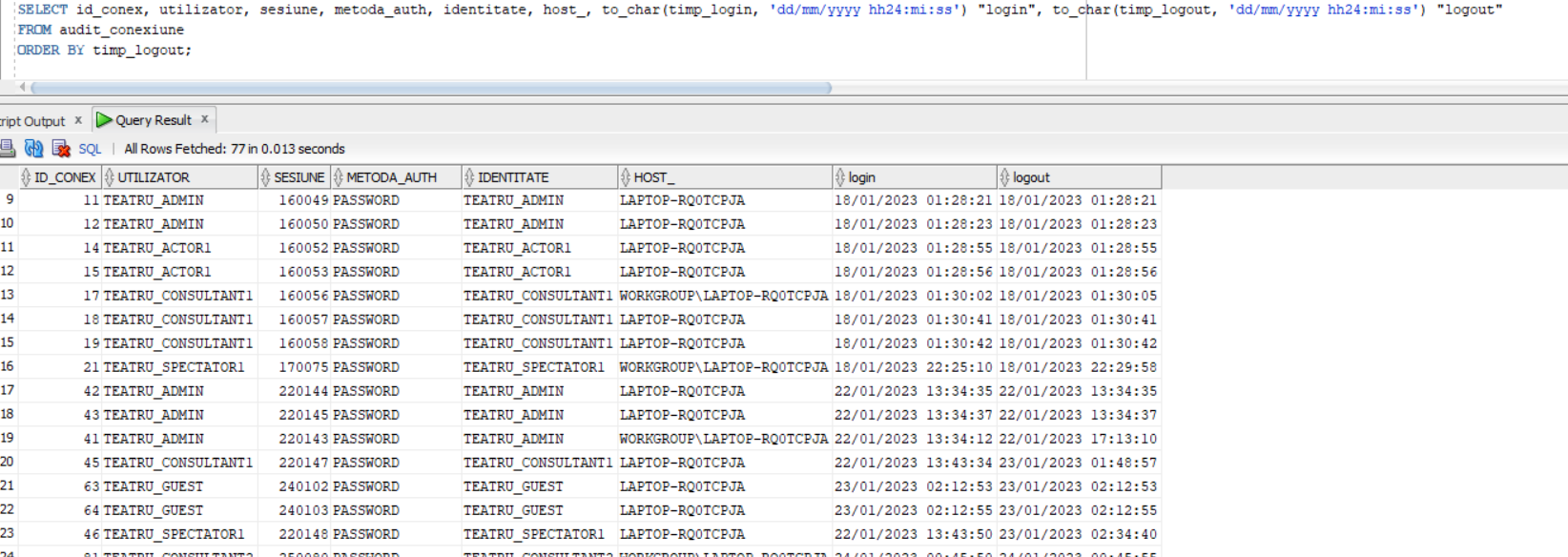
AND sesiune = sys\_context('userenv', 'sessionid');

COMMIT;

END IF;

END;

/



De asemenea, mai creăm un trigger împreună cu tabelul *audit\_teatru\_erori* care va insera în tabel ori de câte ori vor fi rezervate mai multe bilete decât locuri disponibile, un spectator va incerca să rezerve un bilet pentru alt spectator sau să insereze recenzii sub identitatea altui spectator.

CREATE OR REPLACE PACKAGE pkt\_max\_locuri

AS

TYPE t\_idx\_cod\_event IS TABLE OF reprezentatii.reprezentatie\_id%TYPE INDEX BY BINARY\_INTEGER;

TYPE t\_idx\_cod\_client IS TABLE OF spectatori.spectator\_id%TYPE INDEX BY BINARY\_INTEGER;

t\_cod\_event t\_idx\_cod\_event;

t\_cod\_client t\_idx\_cod\_client;

v\_nr\_elem BINARY\_INTEGER := 0;

END pkt\_max\_locuri;

/

-- trigger la nivel de linie

CREATE OR REPLACE TRIGGER trig\_max\_locuri\_linie

BEFORE INSERT OR UPDATE OF num\_locuri, reprezentatie\_id ON bilete

FOR EACH ROW

BEGIN

pkt\_max\_locuri.v\_nr\_elem := pkt\_max\_locuri.v\_nr\_elem + 1;

pkt\_max\_locuri.t\_cod\_event(pkt\_max\_locuri.v\_nr\_elem) := :NEW.reprezentatie\_id;

END;

/

CREATE OR REPLACE PROCEDURE proc\_insert\_audit\_teatru\_erori(msg\_eroare VARCHAR2) AS

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

INSERT INTO audit\_teatru\_erori VALUES(audit\_teatru\_erori\_seq.nextval, SYS\_CONTEXT('USERENV','CURRENT\_USER') ,SYSDATE, msg\_eroare);

COMMIT;

END;

/

-- trigger la nivel de comanda

CREATE OR REPLACE TRIGGER trig\_max\_locuri\_instr

AFTER INSERT OR UPDATE OF num\_locuri, reprezentatie\_id ON bilete

DECLARE

v\_cod\_event reprezentatii.reprezentatie\_id%TYPE;

v\_nr\_seats NUMBER;

v\_max\_seats NUMBER;

BEGIN

FOR i IN 1..pkt\_max\_locuri.v\_nr\_elem LOOP

v\_cod\_event := pkt\_max\_locuri.t\_cod\_event(i);

SELECT SUM(num\_locuri) INTO v\_nr\_seats

FROM bilete

WHERE reprezentatie\_id = v\_cod\_event;

SELECT capacitate INTO v\_max\_seats

FROM sali

JOIN reprezentatii USING(sala\_id)

WHERE reprezentatie\_id = v\_cod\_event;

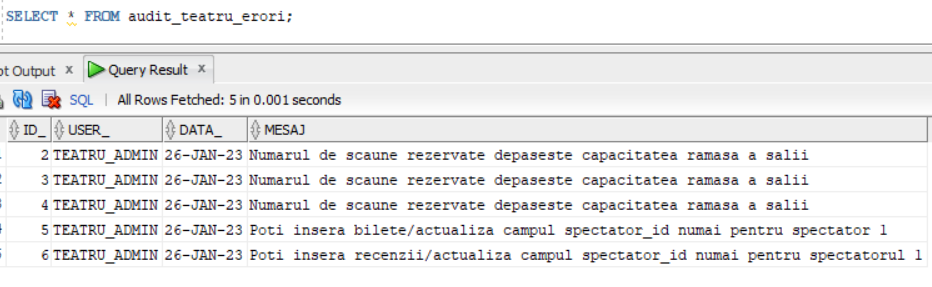
IF v\_nr\_seats > v\_max\_seats THEN

INSERT INTO audit\_teatru\_erori VALUES(audit\_teatru\_erori\_seq.nextval, SYS\_CONTEXT('USERENV','CURRENT\_USER') ,SYSDATE, 'Numarul de scaune rezervate depaseste capacitatea ramasa a salii');

COMMIT; END IF; END LOOP;

pkt\_max\_locuri.v\_nr\_elem := 0;

END; /



* 1. Politici de auditare

Vom face o politică de auditare care inserează în tabelul creat anterior, *audit\_teatru\_erori,* de fiecare dată când un consultant artistic va face update pe tabela reprezentații.

CREATE OR REPLACE PROCEDURE proc\_politica\_auditare AS

BEGIN

dbms\_fga.add\_policy(

object\_schema => 'TEATRU\_ADMIN',

object\_name => 'REPREZENTATII',

policy\_name => 'POLITICA\_UPDATE\_REPREZENTATII',

audit\_condition => 'SYS\_CONTEXT(''USERENV'', ''SESSION\_USER'') LIKE ''%CONSULTANT%''',

enable => false,

statement\_types => 'UPDATE',

handler\_module => 'proc\_auditare\_reprezentatii');

END;

/

CREATE OR REPLACE PROCEDURE proc\_auditare\_reprezentatii (object\_schema varchar2, object\_name varchar2, policy\_name varchar2) AS

PRAGMA AUTONOMOUS\_TRANSACTION;

v\_user VARCHAR2(30) := SYS\_CONTEXT('USERENV','SESSION\_USER');

v\_msg VARCHAR2(255) := 'Politica auditare: utilizatorul ' || v\_user || ' a facut update la o reprezentatie.';

BEGIN

INSERT INTO teatru\_admin.audit\_teatru\_erori VALUES(teatru\_admin.audit\_teatru\_erori\_seq.nextval, v\_user ,SYSDATE, v\_msg);

COMMIT;

END;

/

BEGIN

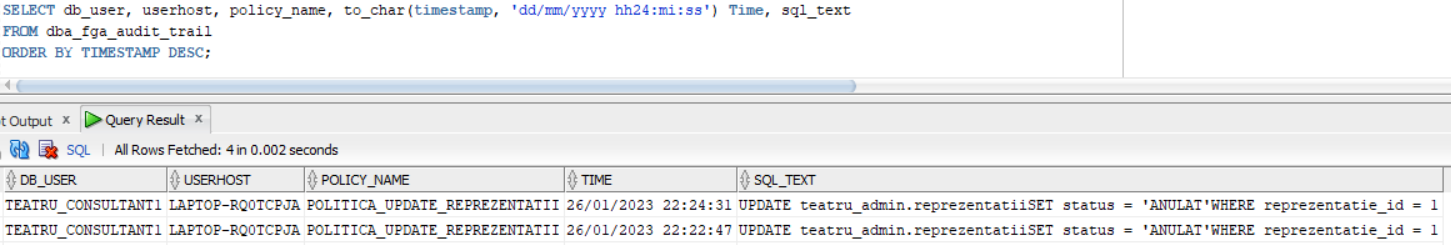
dbms\_fga.enable\_policy(object\_schema => 'TEATRU\_ADMIN',

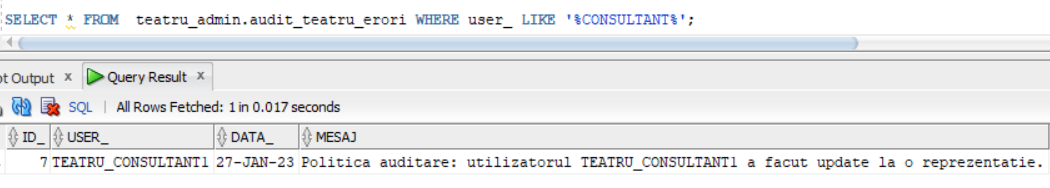
object\_name => 'REPREZENTATII',

policy\_name => 'POLITICA\_UPDATE\_REPREZENTATII');

END;

/

 După ce am adăugat politica de auditare, putem vedea că în tabelul *dba\_fga\_audi\_trail* s-a adăugat operația realizată de consultant.

 De asemenea, helperul creat a adăugat un mesaj de logging în tabela *audit\_teatru\_erori.*

1. Gestiunea utilizatorilor bazei de date și a resurselor computaționale

4.1. Proiectarea configurației de management a identităților în baza de date

Utilizatorii aplicației vor fi:

* Consultanții artistici, care vor administra spectacolele de teatru - CNP
* Actorii, responsabili cu acordarea unui răspuns pentru recenziile lăsate de spectatori la piesa de teatru - CNP
* Administratorul aplicației și al bazei de date - CNP
* Spectatorii, care își vor creea cont pentru a achiziționa bilet - CNP
* Publicul larg, care va accesa aplicația pentru a vedea piesele de teatru – CNP

Singura modalitate de identificare a tuturor utilizatorilor este CNP-ul, însă cum este o informație sensibilă și poate fi aflată, vom folosi pentru identificare pentru toți utilizatorii perechea username – parolă.

Procesele din cadrul aplicației sunt:

1. Crearea piesei de teatru
2. Adăugarea unui teatru, respectiv a unei săli în care pot avea loc piesele de teatru
3. Adăugarea unei reprezentații pentru o piesă de teatru
4. Vizualizarea pieselor de teatru
5. Vizualizarea reprezentațiilor (cu posibilitatea filtrării după dată sau după locație)
6. Achiziționarea biletului la o reprezentație
7. Vizualizarea și modificarea detaliilor de pe bilete
8. Vizualizarea recenziilor date de alți utilizatori la o piesă de teatru
9. Acordarea unei recenzii pentru o piesă de teatru
10. Editarea unei recenzii
11. Acordarea unui răspuns la o recenzie
12. Modificarea datelor profilului unui actor
13. Administrarea utilizatorilor

*Matricea proces – utilizator*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 |
| Consultanții artistici | X | X | X | X | X |  |  | X |  |  |  |  |  |
| Actorii |  |  |  | X | X |  |  | X |  |  | X | X |  |
| Administratorul aplicației | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Spectatorii |  |  |  | X | X | X | X | X | X | X |  |  |  |
| Publicul larg |  |  |  | X | X |  |  | X |  |  |  |  |  |

*Matricea entitate – proces*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 |
| PIESE | I |  | S | S |  |  |  | S | S |  |  |  |  |
| TEATRU |  | S, I |  |  | S |  |  |  |  |  |  |  |  |
| SALI |  | I | S |  | S |  |  |  |  |  |  |  |  |
| REPREZEN-TATII |  |  | I |  | S | S |  |  |  |  |  |  |  |
| BILET |  |  |  |  |  | I | S, U |  |  |  |  |  |  |
| SPECTATORI |  |  |  |  |  |  |  |  |  |  |  |  | I, U |
| ACTORI | S, I |  |  |  |  |  |  |  |  |  |  | U | I, U |
| RECENZII |  |  |  |  |  |  |  | S | I | U | S |  |  |
| RASPUNS |  |  |  |  |  |  |  |  |  |  | I |  |  |
| JOACA | I |  |  |  |  |  |  |  |  |  |  |  |  |

*Matricea entitate – utilizator*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PIESE | TEATRU | SALI | REPREZ. | BILET | SPECTATORI |
| Consultanții artistici | S, I | S, I | S, I | S, I |  |  |
| Actorii | S | S | S | S |  |  |
| Spectatorii | S |  |  | S | S, I, U |  |
| Administratorul aplicației | S, I | S, I | S, I | S, I | S, I, U | I, U |
| Publicul larg | S | S | S | S |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | RECENZII | RASPUNS\_RECENZII | ACTORI | JOACA |
| Consultanții artistici | S |  | S, I | I |
| Actorii | S | I | U |  |
| Spectatorii | S, I, U |  |  |  |
| Administratorul aplicației | S, I, U | I | S, I, U | I |
| Publicul larg | S |  |  |  |

Întrucât procesele nu includ toate operațiile CRUD, pentru a putea defini toate privilegiile utilizatorilor asupra obiectelor, construim și matricea proceselor CRUD:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PIESE | TEATRU | SALI | REPREZ. | BILET | SPECTATORI |
| Consultanții artistici | S, I, U, D | S, I, U, D | S, I, U, D | S, I, U, D | S |  |
| Actorii | S | S | S | S |  |  |
| Spectatorii | S | S | S | S | S (view), I (view), U (view) | S (view) |
| Administratorul aplicației | S, I, U, D | S, I, U, D | S, I, U, D | S, I, U, D | S, I, U | S, I, U, D |
| Publicul larg | S | S | S | S |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | RECENZII | RASPUNS\_RECENZII | ACTORI | JOACA |
| Consultanții artistici | S | S | S (view), I, U, D | S, I, U, D |
| Actorii | S | S, I, U (view), D (view) | S, U (view) | S |
| Spectatorii | S, I, U (view), D (view) | S | S (view) | S |
| Administratorul aplicației | S, I, U, D | S, I, U, D | S, I, U, D | S, I, U, D |
| Publicul larg | S | S | S (view) | S |

4.2. Implementarea configurației de management

În ceea ce privește utilizatorii aplicației, au fost create următoarele conturi:

* Cu autentificare locală:
  + Un cont de tip admin TEATRU\_ADMIN
  + 6 conturi de tip spectator (pentru simplitate) TEATRU\_SPECTATOR<1…6>
  + 5 conturi de tip actor TEATRU\_ACTOR<1…5>
  + 3 conturi de tip consultant artistic TEATRU\_CONSULTANT<1…3>
  + 1 cont de tip guest TEATRU\_GUEST
* Cu autentificare la nivelul sistemului de operare:
  + "OPS$LAPTOP-RQ0TCPJA\COZMA"

CREATE USER "OPS$LAPTOP-RQ0TCPJA\COZMA" IDENTIFIED EXTERNALLY;

GRANT CREATE SESSION TO "OPS$LAPTOP-RQ0TCPJA\COZMA";

GRANT CONNECT TO "OPS$LAPTOP-RQ0TCPJA\COZMA";

-- creare utilizatori locali

-- cream contul de admin

CREATE USER teatru\_admin IDENTIFIED BY pass\_admin PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_admin;

-- cream contul de guest

CREATE USER teatru\_guest IDENTIFIED BY guest;

GRANT CREATE SESSION TO teatru\_guest;

-- cream conturile pentru spectatori

CREATE USER teatru\_spectator1 IDENTIFIED BY pass\_spectator1 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_spectator1;

CREATE USER teatru\_spectator2 IDENTIFIED BY pass\_spectator2 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_spectator2;

CREATE USER teatru\_spectator3 IDENTIFIED BY pass\_spectator3 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_spectator3;

CREATE USER teatru\_spectator4 IDENTIFIED BY pass\_spectator4 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_spectator4;

CREATE USER teatru\_spectator5 IDENTIFIED BY pass\_spectator5 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_spectator5;

CREATE USER teatru\_spectator6 IDENTIFIED BY pass\_spectator6 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_spectator6;

-- cream conturile pentru actori

CREATE USER teatru\_actor1 IDENTIFIED BY pass\_actor1 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor1;

CREATE USER teatru\_actor2 IDENTIFIED BY pass\_actor2 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor2;

CREATE USER teatru\_actor3 IDENTIFIED BY pass\_actor3 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor3;

CREATE USER teatru\_actor4 IDENTIFIED BY pass\_actor4 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor4;

CREATE USER teatru\_actor5 IDENTIFIED BY pass\_actor5 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor5;

-- cream conturile pentru consultantii artistici

CREATE USER teatru\_consultant1 IDENTIFIED BY pass\_consultant1 PASSWORD EXPIRE;

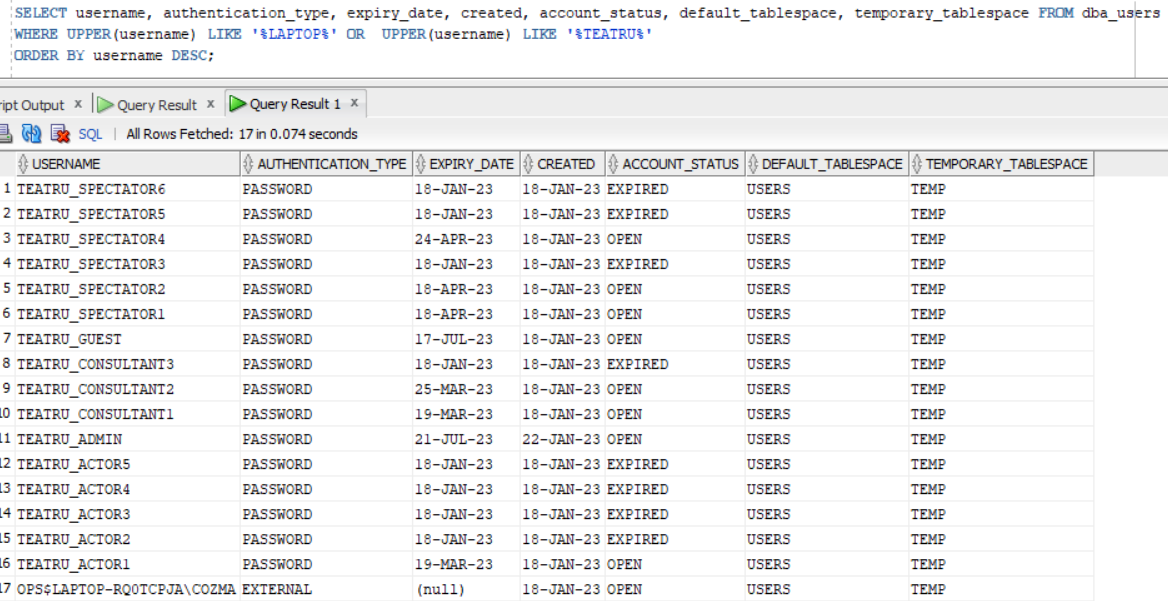
GRANT CREATE SESSION TO teatru\_consultant1;

CREATE USER teatru\_consultant2 IDENTIFIED BY pass\_consultant2 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_consultant2;

CREATE USER teatru\_consultant3 IDENTIFIED BY pass\_consultant3 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_consultant3;



-- cream conturile pentru actori

CREATE USER teatru\_actor1 IDENTIFIED BY pass\_actor1 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor1;

CREATE USER teatru\_actor2 IDENTIFIED BY pass\_actor2 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor2;

CREATE USER teatru\_actor3 IDENTIFIED BY pass\_actor3 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor3;

CREATE USER teatru\_actor4 IDENTIFIED BY pass\_actor4 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor4;

CREATE USER teatru\_actor5 IDENTIFIED BY pass\_actor5 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_actor5;

-- cream conturile pentru consultantii artistici

CREATE USER teatru\_consultant1 IDENTIFIED BY pass\_consultant1 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_consultant1;

CREATE USER teatru\_consultant2 IDENTIFIED BY pass\_consultant2 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_consultant2;

CREATE USER teatru\_consultant3 IDENTIFIED BY pass\_consultant3 PASSWORD EXPIRE;

GRANT CREATE SESSION TO teatru\_consultant3;

Pentru a putea crea obiecte, modificăm quota utilizatorilor pe tabelspace-ul implicit *Users.* Au fost atribuite în modul următor:

* Admin – nelimitat
* Utilizatorul la nivel de sistem de operare – 100M
* Guest – 0M, nu poate crea obiecte
* Consultanți – 10M
* Actori – 5M
* Spectatori – 3M

ALTER USER TEATRU\_ADMIN QUOTA UNLIMITED ON USERS;

ALTER USER "OPS$LAPTOP-RQ0TCPJA\COZMA" QUOTA 100M ON USERS;

ALTER USER TEATRU\_GUEST QUOTA 0M ON USERS;

ALTER USER TEATRU\_CONSULTANT1 QUOTA 10M ON USERS;

ALTER USER TEATRU\_CONSULTANT2 QUOTA 10M ON USERS;

ALTER USER TEATRU\_CONSULTANT3 QUOTA 10M ON USERS;

GRANT CONNECT TO "OPS$LAPTOP-RQ0TCPJA\COZMA";

ALTER USER TEATRU\_ACTOR2 QUOTA 5M ON USERS;

ALTER USER TEATRU\_ACTOR3 QUOTA 5M ON USERS;

ALTER USER TEATRU\_ACTOR4 QUOTA 5M ON USERS;

ALTER USER TEATRU\_ACTOR5 QUOTA 5M ON USERS;

ALTER USER TEATRU\_SPECTATOR1 QUOTA 3M ON USERS;

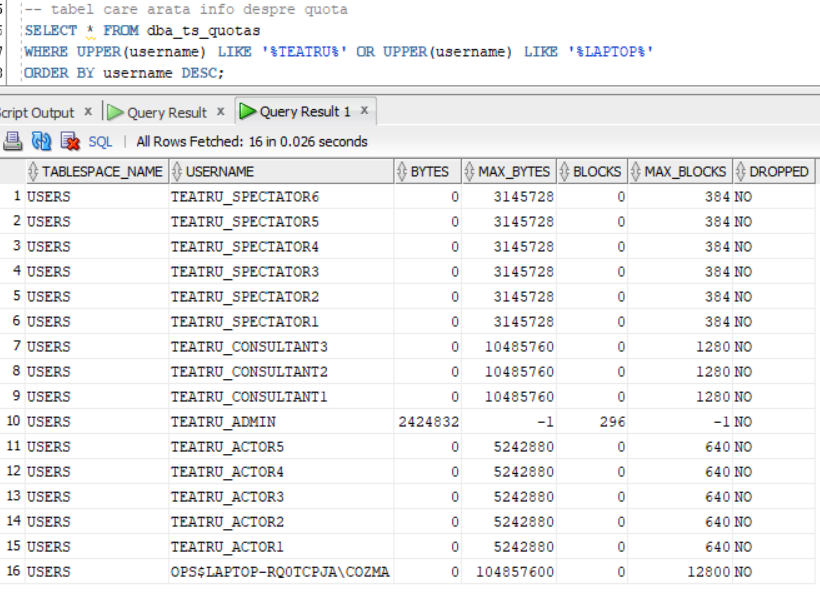
ALTER USER TEATRU\_SPECTATOR2 QUOTA 3M ON USERS;

ALTER USER TEATRU\_SPECTATOR3 QUOTA 3M ON USERS;

ALTER USER TEATRU\_SPECTATOR4 QUOTA 3M ON USERS;

ALTER USER TEATRU\_SPECTATOR5 QUOTA 3M ON USERS;

ALTER USER TEATRU\_SPECTATOR6 QUOTA 3M ON USERS;



Am creat o procedură care configurează un plan de resurse ce conține:

* Un plan de consum, care va fi planul de resurse al aplicației *teatru\_plan\_consum*
* Acesta este împărțit în:
  + Grupul de consum *administratie*, MGMT\_P1 20%, format din TEATRU\_ADMIN și OPS$LAPTOP-RQ0TCPJA\COZMA
  + Un subplan de consum, denumit *teatru\_angajati\_plan\_consum,* ce va folosi MGMT\_P1 30%
    - La rândul lui, acesta va conține 2 grupuri de consum, *consultanti* MGMT\_P1 70% (format din utilizatorii ce vor fi consultanti artistici) și *actori* MGMT\_P1 30% (format din utilizatorii actori)
  + Grupul de consum *public*, format din spectatori si guest, având configurația:
    - MGMT\_P1 45%
    - MAX\_EST\_EXEC\_TIME = 60
    - MAX\_IDLE\_TIME = 600
  + OTHER GROUPS 5%

CREATE OR REPLACE PROCEDURE teatru\_plan\_consum AS

n NUMBER :=0;

BEGIN

-- cream o zona de lucru pentru definirea planului

DBMS\_RESOURCE\_MANAGER.CREATE\_PENDING\_AREA();

-- cream planul de consum

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN(PLAN => 'teatru\_plan\_consum',

COMMENT => 'Planul de consum pentru aplicatia de achizitionare de bilete la teatru');

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN(PLAN => 'teatru\_angajati\_plan\_consum',

COMMENT => 'Planul de consum pentru angajatii (consultanti artistici sau actori) ai teatrului');

-- crearea grupurilor de consum

DBMS\_RESOURCE\_MANAGER.CREATE\_CONSUMER\_GROUP(CONSUMER\_GROUP => 'administratie', COMMENT => 'Grupeaza sesiunile administratorilor aplicatiei');

DBMS\_RESOURCE\_MANAGER.CREATE\_CONSUMER\_GROUP(CONSUMER\_GROUP => 'consultanti', COMMENT => 'Grupeaza sesiunile consultantilor artistici');

DBMS\_RESOURCE\_MANAGER.CREATE\_CONSUMER\_GROUP(CONSUMER\_GROUP => 'actori', COMMENT => 'Grupeaza sesiunile actorilor');

DBMS\_RESOURCE\_MANAGER.CREATE\_CONSUMER\_GROUP(CONSUMER\_GROUP => 'public', COMMENT => 'Grupeaza sesiunile potentialilor spectatori si ale vizitatorilor aplicatiei');

-- crearea grupului de consum pentru utilizatorii din categoria others

SELECT COUNT(\*) INTO n

FROM DBA\_RSRC\_CONSUMER\_GROUPS

WHERE CONSUMER\_GROUP='OTHER\_GROUPS';

IF n=0 THEN

DBMS\_RESOURCE\_MANAGER.CREATE\_CONSUMER\_GROUP(CONSUMER\_GROUP => 'OTHER\_GROUPS', COMMENT => 'Grupeaza restul utilizatorilor');

END IF;

--mapari statice utilizatori pe grupuri consum, nu pot fi mapati pe grupul OTHER\_GROUPS

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ADMIN', 'administratie');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'OPS$LAPTOP-RQ0TCPJA\COZMA', 'administratie');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_CONSULTANT1', 'consultanti');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_CONSULTANT2', 'consultanti');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_CONSULTANT3', 'consultanti');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR1', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR2', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR3', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR4', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR5', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR1', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR2', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR3', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR4', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR5', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR6', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_GUEST', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'OPS$LAPTOP-RQ0TCPJA\COZMA', 'administratie');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_CONSULTANT1', 'consultanti');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_CONSULTANT2', 'consultanti');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_CONSULTANT3', 'consultanti');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR1', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR2', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR3', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR4', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_ACTOR5', 'actori');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR1', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR2', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR3', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR4', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR5', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_SPECTATOR6', 'public');

DBMS\_RESOURCE\_MANAGER.SET\_CONSUMER\_GROUP\_MAPPING(DBMS\_RESOURCE\_MANAGER.ORACLE\_USER, 'TEATRU\_GUEST', 'public');

--directivele de plan pentru fiecare grup de consum

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN\_DIRECTIVE(PLAN => 'teatru\_plan\_consum', GROUP\_OR\_SUBPLAN => 'administratie',

COMMENT => 'directiva plan pentru administratie', MGMT\_P1 => 20);

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN\_DIRECTIVE(PLAN => 'teatru\_plan\_consum', GROUP\_OR\_SUBPLAN => 'teatru\_angajati\_plan\_consum',

COMMENT => 'directiva plan pentru angajati', MGMT\_P1 => 30);

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN\_DIRECTIVE(PLAN => 'teatru\_angajati\_plan\_consum', GROUP\_OR\_SUBPLAN => 'consultanti',

COMMENT => 'directiva plan pentru consultanti artistici', MGMT\_P1 => 70);

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN\_DIRECTIVE(PLAN => 'teatru\_angajati\_plan\_consum', GROUP\_OR\_SUBPLAN => 'actori',

COMMENT => 'directiva plan pentru consultanti artistici', MGMT\_P1 => 30);

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN\_DIRECTIVE(PLAN => 'teatru\_plan\_consum', GROUP\_OR\_SUBPLAN => 'public',

COMMENT => 'directiva plan pentru spectatori/guest', MGMT\_P1 => 45, MAX\_EST\_EXEC\_TIME => 60, MAX\_IDLE\_TIME => 600);

DBMS\_RESOURCE\_MANAGER.CREATE\_PLAN\_DIRECTIVE(PLAN => 'teatru\_plan\_consum', GROUP\_OR\_SUBPLAN => 'OTHER\_GROUPS',

COMMENT => 'directiva plan alti utilizatori', MGMT\_P1 => 5);

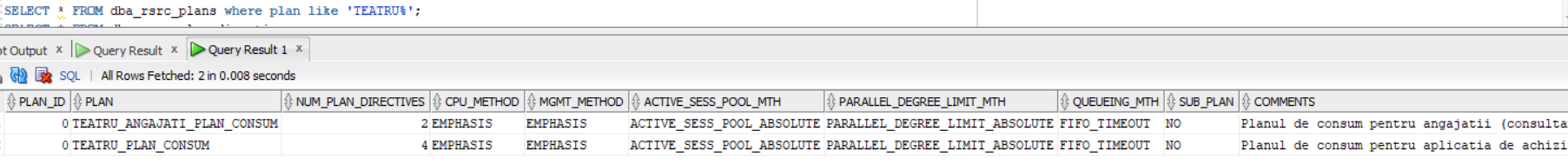
DBMS\_RESOURCE\_MANAGER.VALIDATE\_PENDING\_AREA();

DBMS\_RESOURCE\_MANAGER.SUBMIT\_PENDING\_AREA();

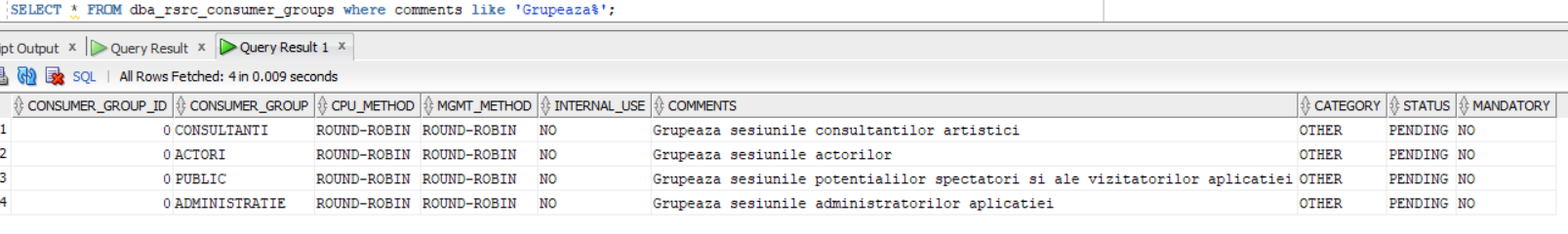
END;

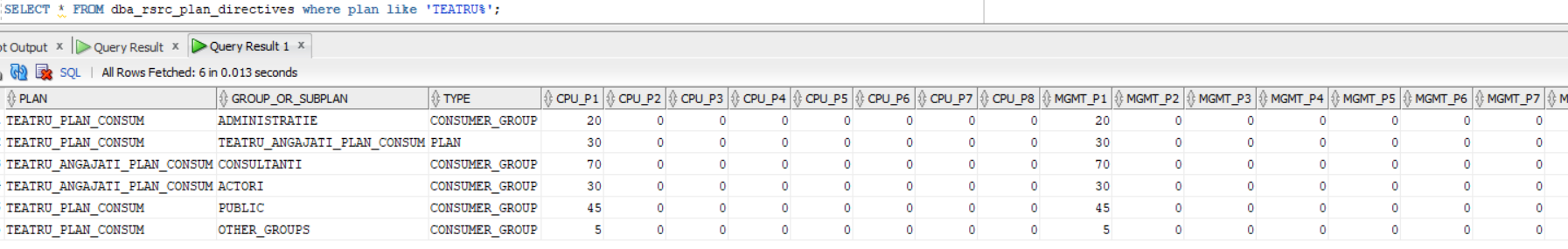
/

Cele două planuri de consum:



Grupurile de consum:



Directivele planurilor de consum:

Creăm 3 tipuri de profil de utilizator:

* *Teatru\_guest* (pentru guest), care poate avea maxim 5 sesiuni per user și maxim 5 încercări de login cu eșec
* *Teatru\_spectator* (pentru spectator)cu*:*
  + SESSIONS\_PER\_USER 5
  + FAILED\_LOGIN\_ATTEMPTS 5
  + PASSWORD\_LIFE\_TIME 90
  + PASSWORD\_REUSE\_TIME 60
  + PASSWORD\_REUSE\_MAX 5
* *Teatru\_angajat* (pentru admin, utilizatorul SO, actori și consultanți) cu:
  + SESSIONS\_PER\_USER 5
  + FAILED\_LOGIN\_ATTEMPTS 3
  + PASSWORD\_LIFE\_TIME 60
  + PASSWORD\_REUSE\_TIME 60
  + PASSWORD\_REUSE\_MAX 3

CREATE PROFILE teatru\_guest LIMIT

SESSIONS\_PER\_USER 5

FAILED\_LOGIN\_ATTEMPTS 5;

CREATE PROFILE teatru\_spectator LIMIT

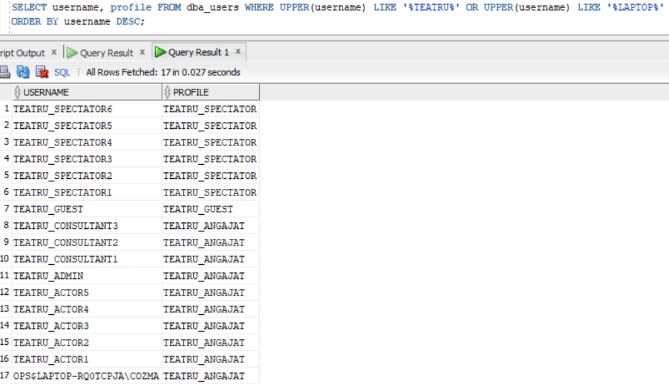
SESSIONS\_PER\_USER 5

FAILED\_LOGIN\_ATTEMPTS 5

PASSWORD\_LIFE\_TIME 90

PASSWORD\_REUSE\_TIME 60

PASSWORD\_REUSE\_MAX 5;



CREATE PROFILE teatru\_angajat LIMIT

SESSIONS\_PER\_USER 5

FAILED\_LOGIN\_ATTEMPTS 3

PASSWORD\_LIFE\_TIME 60

PASSWORD\_REUSE\_TIME 60

PASSWORD\_REUSE\_MAX 3;

ALTER USER "OPS$LAPTOP-RQ0TCPJA\COZMA" PROFILE teatru\_angajat;

ALTER USER TEATRU\_ACTOR1 PROFILE teatru\_angajat;

ALTER USER TEATRU\_ACTOR2 PROFILE teatru\_angajat;

ALTER USER TEATRU\_ACTOR3 PROFILE teatru\_angajat;

ALTER USER TEATRU\_ACTOR4 PROFILE teatru\_angajat;

ALTER USER TEATRU\_ACTOR5 PROFILE teatru\_angajat;

ALTER USER TEATRU\_ADMIN PROFILE teatru\_angajat;

ALTER USER TEATRU\_CONSULTANT1 PROFILE teatru\_angajat;

ALTER USER TEATRU\_CONSULTANT2 PROFILE teatru\_angajat;

ALTER USER TEATRU\_CONSULTANT3 PROFILE teatru\_angajat;

ALTER USER TEATRU\_SPECTATOR1 PROFILE teatru\_spectator;

ALTER USER TEATRU\_SPECTATOR2 PROFILE teatru\_spectator;

ALTER USER TEATRU\_SPECTATOR3 PROFILE teatru\_spectator;

ALTER USER TEATRU\_SPECTATOR4 PROFILE teatru\_spectator;

ALTER USER TEATRU\_SPECTATOR5 PROFILE teatru\_spectator;

ALTER USER TEATRU\_SPECTATOR6 PROFILE teatru\_spectator;

ALTER USER TEATRU\_GUEST PROFILE teatru\_guest;

1. Privilegii și roluri

5.1 Privilegii sistem și obiect

* Întrucât în cadrul operației de select pe tabela ACTORI, utilizatorii de tip consultant artistic, spectator, actori (își va putea consulta numai datele personale) și publicul larg nu ar trebui să aibă dreptul de a vedea anumite informații referitoare la actori (username, parola, email, numărul de telefon), vom crea o *vizualizare* view\_actori ce conține doar informațiile selectate. De asemenea, pentru simplitate, vom crea *rolul* vizualizare\_actori pe care îl vom atribui utilizatorilor enumerați mai sus.

CREATE OR REPLACE VIEW view\_actori AS SELECT actor\_id, prenume, nume, locul\_nasterii, data\_nasterii, biografie FROM actori;

SELECT \* FROM view\_actori;

-- Cream vizualizare si oferim utilizatorilor drepturi de read pe vizualizare

CREATE ROLE vizualizare\_actori;

GRANT SELECT ON view\_actori to vizualizare\_actori;

GRANT vizualizare\_actori TO TEATRU\_CONSULTANT1;

GRANT vizualizare\_actori TO TEATRU\_CONSULTANT2;

GRANT vizualizare\_actori TO TEATRU\_CONSULTANT3;

GRANT vizualizare\_actori TO TEATRU\_ACTOR1;

GRANT vizualizare\_actori TO TEATRU\_ACTOR2;

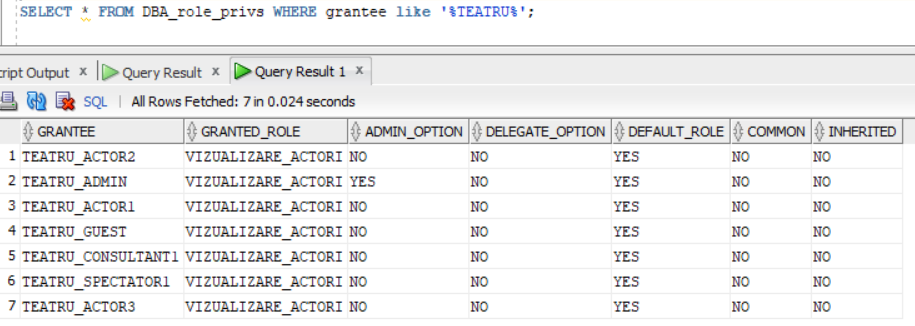
GRANT vizualizare\_actori TO TEATRU\_ACTOR3;

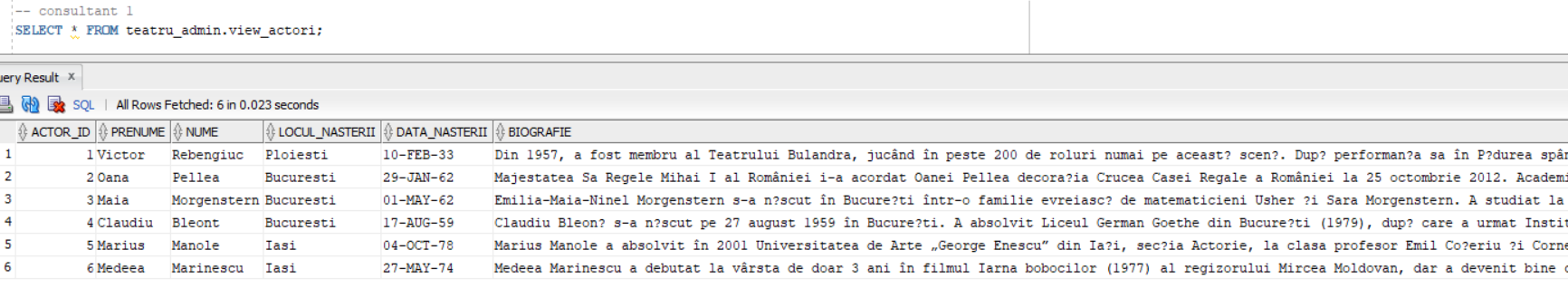
GRANT vizualizare\_actori TO TEATRU\_SPECTATOR1;

GRANT vizualizare\_actori TO TEATRU\_SPECTATOR2;

GRANT vizualizare\_actori TO TEATRU\_SPECTATOR3;

GRANT vizualizare\_actori TO TEATRU\_GUEST;

Rolurile nou atribuite pot fi vazute în tabela DBA\_ROLE\_PRIVS:

 Acum utilizatorii au drept de citire al datelor din vizualizare, astfel ca vom încerca să facem un select în cadrul sesiunii utilizatorului TEATRU\_CONSULTANT1:

* Definim rolul *rol\_consultant* care va conține toate drepturile oferite consultanților artistici. Acum consultantul 2 va putea face select, update, insert și delete pe PIESE\_TEATRU, însă nu va putea face altă operație decât select RECENZII.

CREATE ROLE rol\_consultant;

GRANT SELECT, UPDATE, DELETE, INSERT ON piese\_teatru TO rol\_consultant;

GRANT SELECT, UPDATE, DELETE, INSERT ON joaca TO rol\_consultant;

GRANT SELECT, UPDATE, DELETE, INSERT ON teatru TO rol\_consultant;

GRANT SELECT, UPDATE, DELETE, INSERT ON sali TO rol\_consultant;

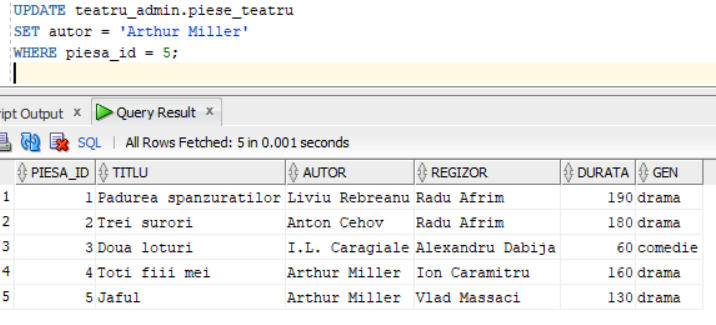
GRANT SELECT, UPDATE, DELETE, INSERT ON reprezentatii TO rol\_consultant;

GRANT SELECT, UPDATE, DELETE, INSERT ON view\_actori TO rol\_consultant;

GRANT SELECT ON bilete TO rol\_consultant;

GRANT SELECT ON recenzii TO rol\_consultant;

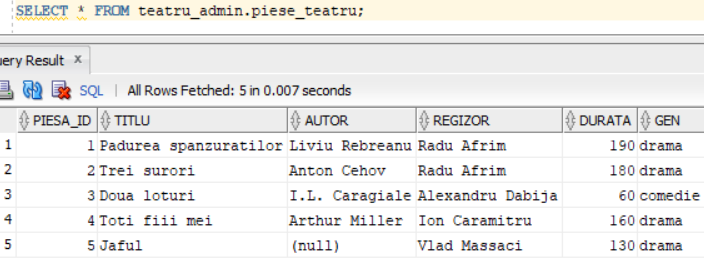
GRANT SELECT ON raspuns\_recenzii TO rol\_consultant;

Actualizăm câmpul autor pentru piesa 5.

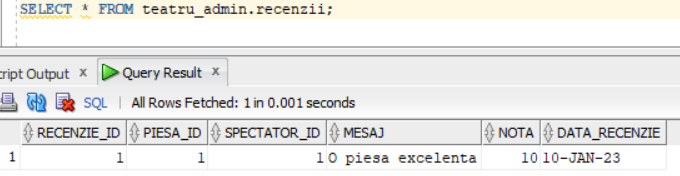
GRANT rol\_consultant TO TEATRU\_CONSULTANT1;

GRANT rol\_consultant TO TEATRU\_CONSULTANT2;

GRANT rol\_consultant TO TEATRU\_CONSULTANT3;

 Selectăm piesele de teatru.

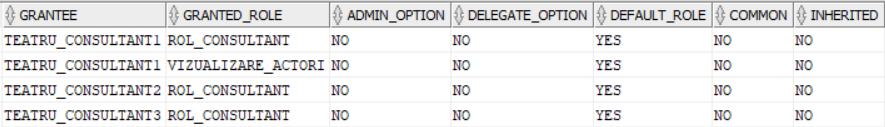
Select vs. update pe recenzii de către un consultant:



Operația de select este realizată cu succes.



Update-ul nu e posibil pe recenzii.

 Drepturile *consultantilor*:

* Se observă faptul că actorii, spectatorii și publicul larg au numai drept de select pe tabelele: PIESE, JOACA, TEATRU, SALI, REPREZENTATII, RECENZII, RASPUNS\_RECENZII. Vom crea un *rol* care să cuprindă toate privilegiile pe tabelele respective.

CREATE ROLE vizualizare\_tabele\_public;

GRANT SELECT ON piese\_teatru TO vizualizare\_tabele\_public;

GRANT SELECT ON joaca TO vizualizare\_tabele\_public;

GRANT SELECT ON teatru TO vizualizare\_tabele\_public;

GRANT SELECT ON sali TO vizualizare\_tabele\_public;

GRANT SELECT ON reprezentatii TO vizualizare\_tabele\_public;

GRANT SELECT ON recenzii TO vizualizare\_tabele\_public;

GRANT SELECT ON raspuns\_recenzii TO vizualizare\_tabele\_public;

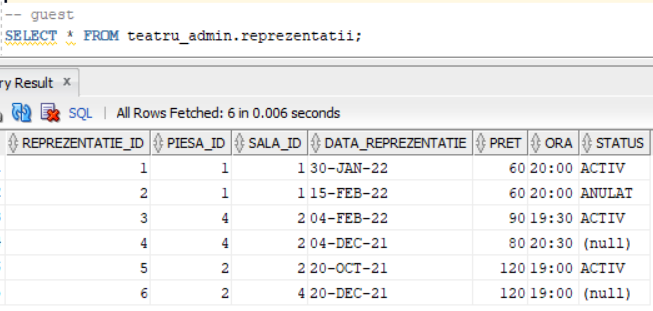
GRANT vizualizare\_tabele\_public TO teatru\_guest;

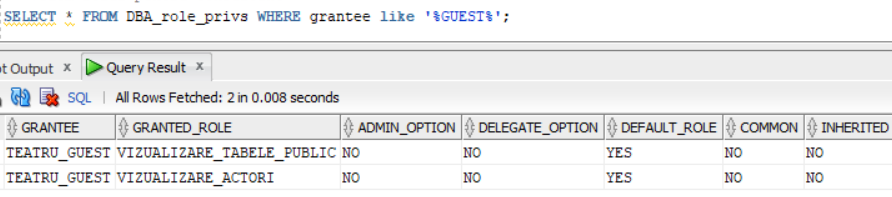
GRANT vizualizare\_tabele\_public TO TEATRU\_ACTOR1;

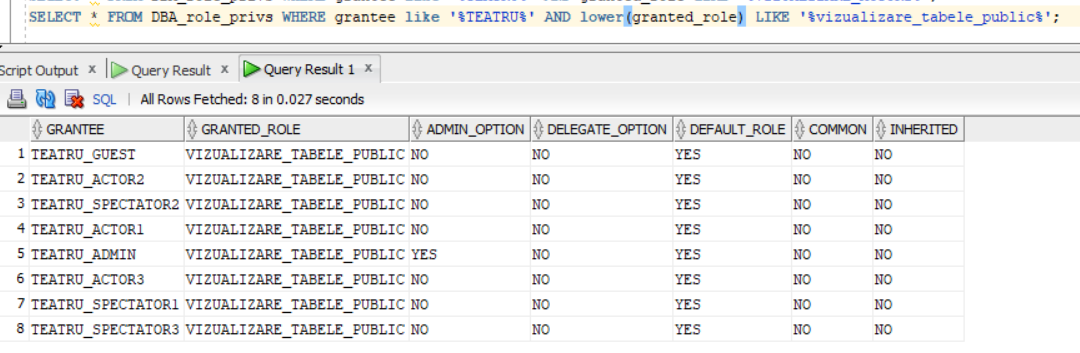
GRANT vizualizare\_tabele\_public TO TEATRU\_SPECTATOR1;

GRANT vizualizare\_tabele\_public TO TEATRU\_SPECTATOR2;

GRANT vizualizare\_tabele\_public TO TEATRU\_SPECTATOR3;

 Dacă vom intra în contul guest, vom putea vedea că putem face cereri select pe una din tabele.

Acum putem vedea toate drepturile contului guest:

 Drepturile oferite către *vizualizare\_tabele\_public*:

Continuăm cu drepturile pentru utilizatorii de tip *spectator.*

* Cum fiecare spectator își va putea modifica doar biletele proprii, vom crea în schema adminului câte o vizualizare cu biletele corespunzatoare corespunzătoare pentru fiecare .

CREATE OR REPLACE VIEW view\_bilete1 AS SELECT \* FROM bilete WHERE spectator\_id = 1;

GRANT SELECT, UPDATE, INSERT, DELETE ON TEATRU\_ADMIN.view\_bilete1 TO TEATRU\_SPECTATOR1;

Pentru ca schimbările din view să fie reflectate și în tabelul de bază, vom crea tiggeri INSTEAD OF pentru fiecare view de bilete corespunzător spectatorilor. Atunci când inserăm în vizualizare biletul (2, 1, 2), va fi inserat și în tabelul bilete.

De asemenea, un spectator nu va putea crea bilete pentru alt spectator și nici nu va putea actualiza coloana *spectator\_id* cu altă valoare decât a sa.

CREATE OR REPLACE TRIGGER trigger\_bilete1

INSTEAD OF INSERT OR UPDATE ON view\_bilete1

FOR EACH ROW

BEGIN

IF :NEW.spectator\_id != 1 THEN

proc\_insert\_audit\_teatru\_erori('Poti insera bilete/actualiza campul spectator\_id numai pentru spectator 1');

RAISE\_APPLICATION\_ERROR(-20001, 'Poti insera bilete/actualiza campul spectator\_id numai pentru spectator 1');

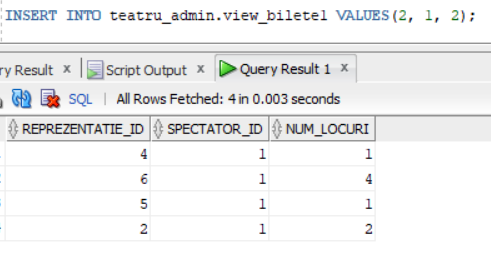
ELSE

INSERT INTO bilete VALUES(:NEW.reprezentatie\_id, :NEW.spectator\_id, :NEW.num\_locuri);

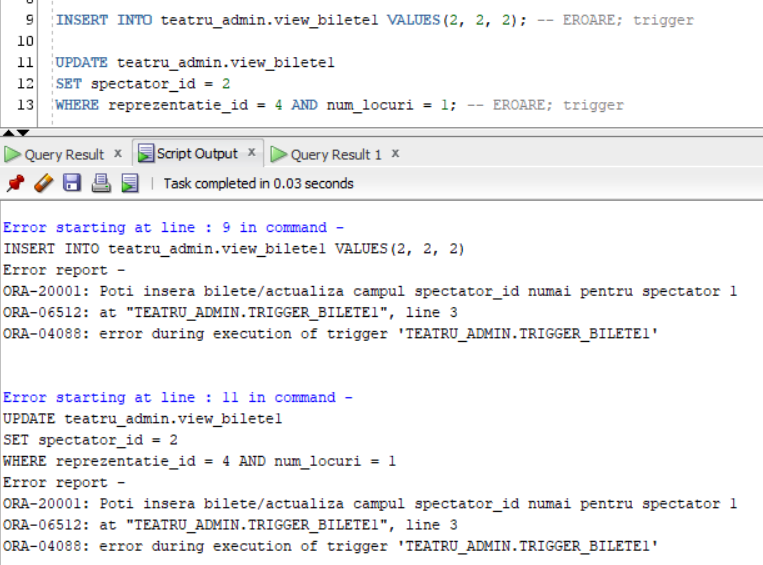
COMMIT;

END IF;

END; /



Vom primi eroare dacă încercăm să actualizăm sau inserăm bilete pentru alt spectator.



* De asemenea, vrem ca *spectatorii*, respectiv *actorii* să poată face UPDATE, INSERT și DELETE numai pe propriile recenzii, respectiv răspunsuri la recenzii. Astfel vom crea vizualizări separate pentru fiecare spectator, respectiv actor, în schema admin.

În plus, adăugăm câte un trigger pe fiecare vizualizare, pentru ca spectatorii și actorii să nu poată insera sau face update la recenzii, respectiv raspunsurile recenziilor, cu id-ul altui spectator.

*Recenzii:*

CREATE OR REPLACE VIEW view\_recenzii1 AS SELECT \* FROM recenzii WHERE spectator\_id = 1;

GRANT SELECT, UPDATE, INSERT, DELETE ON TEATRU\_ADMIN.view\_recenzii1 TO TEATRU\_SPECTATOR1;

SELECT \* FROM view\_recenzii1;

GRANT SELECT ON recenzii\_seq TO TEATRU\_SPECTATOR1;

CREATE OR REPLACE TRIGGER trigger\_recenzii1

INSTEAD OF INSERT OR UPDATE ON view\_recenzii1

FOR EACH ROW

BEGIN

IF :NEW.spectator\_id != 1 THEN

proc\_insert\_audit\_teatru\_erori('Poti insera recenzii/actualiza campul spectator\_id numai pentru spectatorul 1');

RAISE\_APPLICATION\_ERROR(-20011, 'Poti insera recenzii/actualiza campul spectator\_id numai pentru spectatorul 1');

ELSIF (INSERTING) THEN

INSERT INTO recenzii VALUES(:NEW.recenzie\_id, :NEW.piesa\_id, :NEW.spectator\_id, :NEW.mesaj, :NEW.nota, :NEW.data\_recenzie);

COMMIT;

ELSE

UPDATE recenzii

SET piesa\_id = :NEW.piesa\_id, spectator\_id = :NEW.spectator\_id,

mesaj = :NEW.mesaj, nota = :NEW.nota, data\_recenzie = :NEW.data\_recenzie

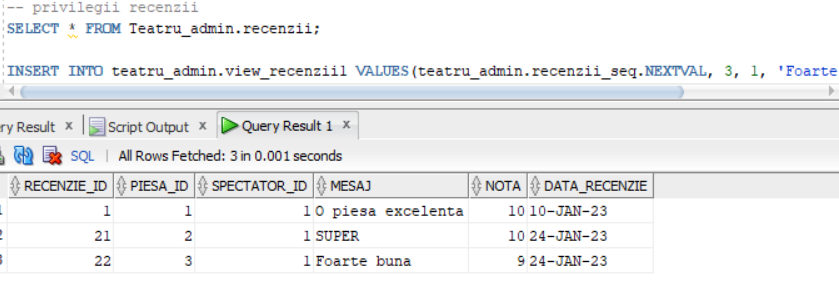
WHERE recenzie\_id = :NEW.recenzie\_id;

COMMIT;

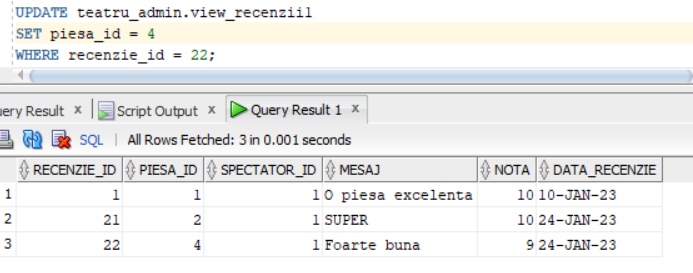
END IF;

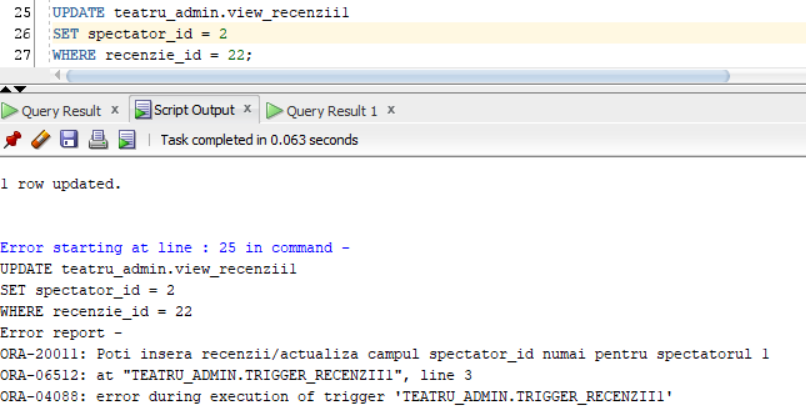
END;

/

Select din spectator1 pe tabela RECENZII după insert în vizualizarea asociată:

Update din spectator1 pentru *piesa\_id* și select pe întreaga tabelă:



Nu se poate face update în cazul în care *spectator\_id* are altă valoare decât 1:

Vom folosi aceeași metodă pentru *răspunsurile la recenzii* pentru ACTORI:

CREATE OR REPLACE VIEW view\_raspuns1 AS SELECT \* FROM raspuns\_recenzii WHERE actor\_id = 1;

GRANT SELECT, UPDATE, INSERT, DELETE ON TEATRU\_ADMIN.view\_raspuns1 TO TEATRU\_ACTOR1;

GRANT SELECT ON raspuns\_seq TO TEATRU\_ACTOR1;

CREATE OR REPLACE TRIGGER trigger\_raspuns1

INSTEAD OF INSERT OR UPDATE ON view\_raspuns1

FOR EACH ROW

BEGIN

IF :NEW.actor\_id != 1 THEN

proc\_insert\_audit\_teatru\_erori('Poti insera raspunsul/actualiza campul actor\_id numai pentru actor 1');

RAISE\_APPLICATION\_ERROR(-20021, 'Poti insera raspunsul/actualiza campul actor\_id numai pentru actor 1');

ELSIF (INSERTING) THEN

INSERT INTO raspuns\_recenzii VALUES(:NEW.raspuns\_id, :NEW.recenzie\_id, :NEW.actor\_id, :NEW.mesaj, :NEW.data\_recenzie);

COMMIT;

ELSE

UPDATE raspuns\_recenzii

SET recenzie\_id = :NEW.recenzie\_id, actor\_id = :NEW.actor\_id,

mesaj = :NEW.mesaj, data\_recenzie = :NEW.data\_recenzie

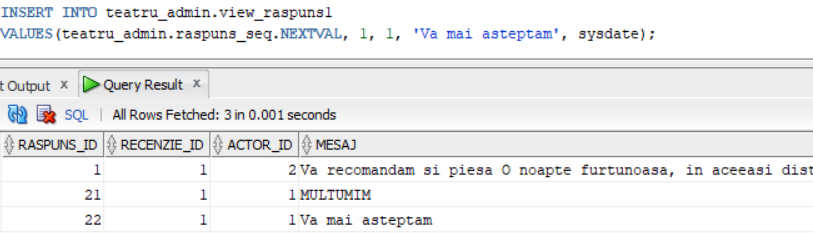
WHERE raspuns\_id = :NEW.raspuns\_id;

COMMIT;

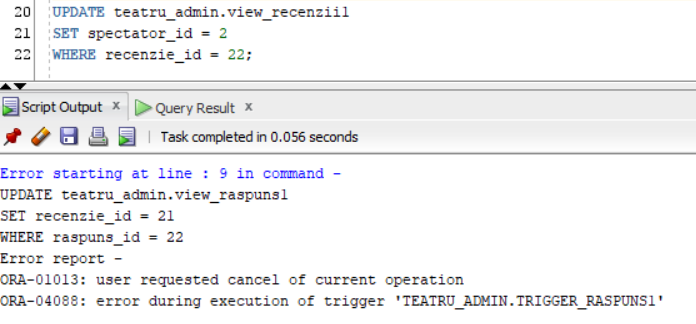
END IF;

END;

/

 Putem insera date:

Nu putem face update la *actor\_id*  pentru un actor diferit:



* Deoarece nu vrem să dăm drepturi de editare a datelor personale ale spectatorilor numai persoanei care e referită, vom face câte un *view* pentru fiecare *spectator*, iar fiecare view va avea asociat un trigger INSTEAD OF, care, la operația *update*, va propaga schimbările în tabela SPECTATORI.

CREATE OR REPLACE VIEW view\_spectator1 AS SELECT spectator\_id, username, prenume, nume

FROM spectatori WHERE spectator\_id = 1;

GRANT SELECT, UPDATE, INSERT, DELETE ON TEATRU\_ADMIN.view\_spectator1 TO TEATRU\_SPECTATOR1;

SELECT \* FROM view\_spectator1;

CREATE OR REPLACE TRIGGER trigger\_spectator1

INSTEAD OF UPDATE ON view\_spectator1

FOR EACH ROW

BEGIN

UPDATE spectatori

SET username = :NEW.username,

prenume = :NEW.prenume,

nume = :NEW.nume

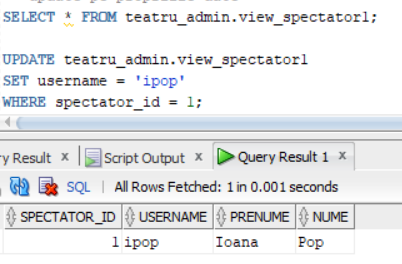
WHERE spectator\_id = 1;

COMMIT;

END;

/

Update pe date:



Similar ca în cazul spectatorului, și *actorul* va putea să își modifice datele proprii. Vom crea vizualizarea *view\_actor1* împreună cu triggerul  *trigger\_actor1.*

CREATE OR REPLACE VIEW view\_actor1 AS SELECT actor\_id, username, prenume, nume, locul\_nasterii, data\_nasterii, biografie

GRANT SELECT, UPDATE, INSERT, DELETE ON TEATRU\_ADMIN.view\_actor1 TO TEATRU\_ACTOR1;

CREATE OR REPLACE TRIGGER trigger\_actor1

INSTEAD OF UPDATE ON view\_actor1

FOR EACH ROW

BEGIN

UPDATE actori

SET username = :NEW.username,

prenume = :NEW.prenume,

nume = :NEW.nume,

locul\_nasterii = :NEW.locul\_nasterii,

data\_nasterii = :NEW.data\_nasterii,

biografie = :NEW.biografie

WHERE actor\_id = 1;

COMMIT;

END;

/

UPDATE actori

SET username = :NEW.username,

prenume = :NEW.prenume,

nume = :NEW.nume,

locul\_nasterii = :NEW.locul\_nasterii,

data\_nasterii = :NEW.data\_nasterii,

biografie = :NEW.biografie

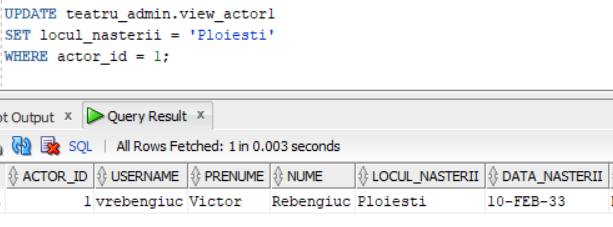
WHERE actor\_id = 1;

COMMIT;

END;

/

Dacă vrem să facem update *locului\_nasterii,* operația este propagată:

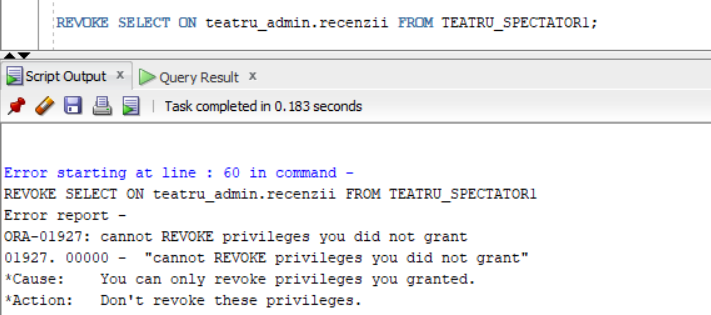


5.2. Ierarhii de privilegii

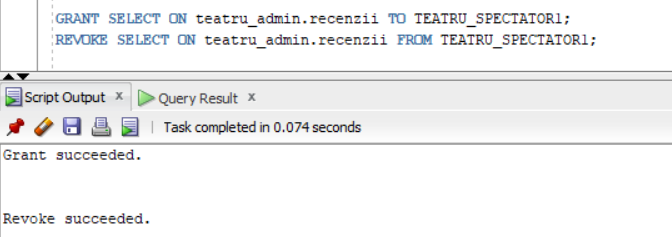
În acest moment, un utilizator de tip spectator are urmatoarele privilegii asupra tabelei *RECENZII* sau a *vizualizărilor* derivate din această tabelă. Acestea sunt:

* SELECT pe RECENZII prin *vizualizare\_tabele\_public*
* SELECT, INSERT, DELETE, UPDATE asupra vizualizării *view\_recenzii1,* ce conține doar recenziile scrise de el

Presupunem că nu mai vrem ca un spectator să aibă privilegiu de select asupra tabelei. Încercăm să scoatem privilegiul prin revoke și primim eroare:



Pentru a putea face acest lucru, trebuie să îi atribuim și separat privilegiu de SELECT, nu numai din interiorul rolului.



5.3. Privilegii asupra obiectelor dependente

Presupunem că vrem ca utilizatorul de tip *guest* să poată aibă privilegiu de select asupra acestei vizualizări:

CREATE OR REPLACE VIEW view\_detalii\_piese AS (SELECT p.piesa\_id, titlu, autor, regizor, durata, gen, COUNT(r.reprezentatie\_id) nr\_reprezentatii, SUM(num\_locuri) nr\_locuri\_vandute FROM piese\_teatru p

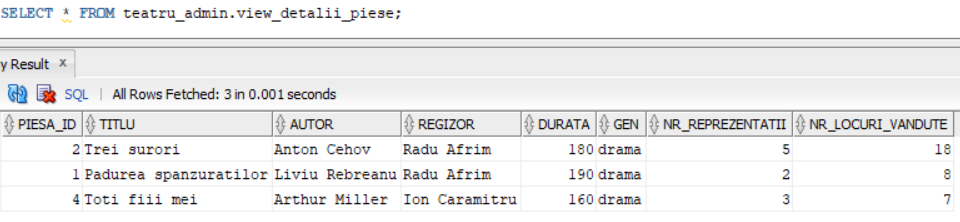
LEFT JOIN reprezentatii r ON (r.piesa\_id = p.piesa\_id)

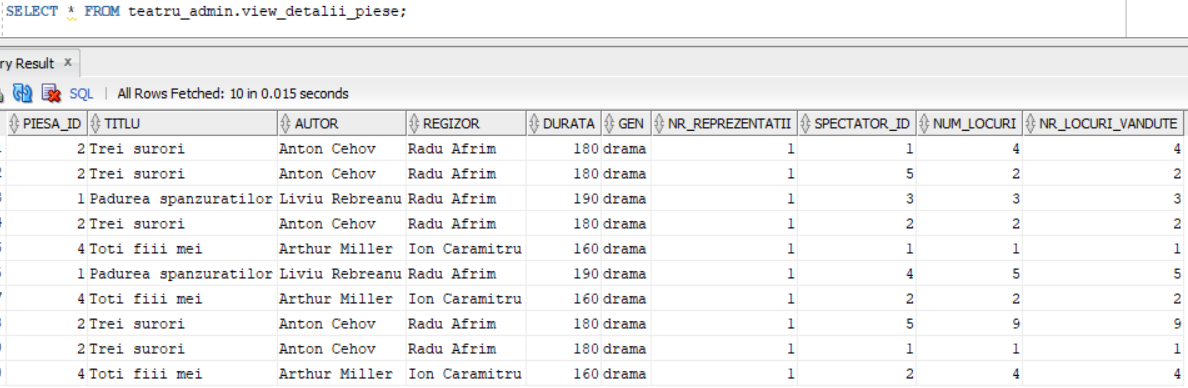
JOIN bilete b ON (b.reprezentatie\_id = r.reprezentatie\_id)

GROUP BY p.piesa\_id, titlu, autor, regizor, durata, gen);

GRANT SELECT ON view\_detalii\_piese TO TEATRU\_GUEST;

GRANT SELECT ON view\_detalii\_piese TO TEATRU\_GUEST;

 Acum utilizatorul *guest* va putea vedea informații despre numărul total de bilete vândute la o piesă de teatru, în ciuda faptului că acesta nu are privilegiu de SELECT asupra tabelei.

 Mai mult decât atât, dacă vizualizarea e modificată astfel încât să expună date din tabelul BILETE, guest le va putea accesa fără grant select pe bilete explicit.

CREATE OR REPLACE VIEW view\_detalii\_piese AS (SELECT p.piesa\_id, titlu, autor, regizor, durata, gen, COUNT(r.reprezentatie\_id) nr\_reprezentatii, b.spectator\_id, b.num\_locuri, SUM(num\_locuri) nr\_locuri\_vandute FROM piese\_teatru p

LEFT JOIN reprezentatii r ON (r.piesa\_id = p.piesa\_id)

JOIN bilete b ON (b.reprezentatie\_id = r.reprezentatie\_id)

GROUP BY p.piesa\_id, titlu, autor, regizor, durata, gen, b.spectator\_id, b.num\_locuri);

GRANT SELECT ON view\_detalii\_piese TO TEATRU\_GUEST;

GRANT SELECT ON view\_detalii\_piese TO TEATRU\_GUEST;GRANT SELECT ON view\_detalii\_piese TO TEATRU\_GUEST;

1. Aplicațiile pe baza de date și securitatea datelor

6.1. Contextul aplicației

Vom adăuga în contextul aplicației la *logon* pentru *admin* și *consultanți* variabilele *permis\_update* și *permis\_delete.*

Acești utilizatori nu vor putea face:

* *Update* dacă suntem în intervalul 12:00 – 23:59
* *Delete* dacă suntem în intervalul 08:00 – 23:59

CREATE CONTEXT teatru\_ctx USING proc\_teatru\_ctx;

CREATE OR REPLACE PROCEDURE proc\_teatru\_ctx IS

v\_ora NUMBER(3);

BEGIN

SELECT TO\_NUMBER(TO\_CHAR(sysdate, 'hh24')) INTO v\_ora

FROM dual;

IF v\_ora >= 8 AND v\_ora <= 23 THEN

dbms\_session.SET\_CONTEXT('teatru\_ctx', 'premis\_delete', 'nu');

ELSE

dbms\_session.SET\_CONTEXT('teatru\_ctx', 'premis\_delete', 'da');

END IF;

IF v\_ora >= 12 AND v\_ora <= 23 THEN

dbms\_session.SET\_CONTEXT('teatru\_ctx', 'premis\_update', 'nu');

ELSE

dbms\_session.SET\_CONTEXT('teatru\_ctx', 'premis\_update', 'da');

END IF;

END; /

EXEC proc\_teatru\_ctx;

CREATE OR REPLACE TRIGGER trigger\_after\_logon\_database

AFTER LOGON ON DATABASE

DECLARE

v\_user varchar2(30);

BEGIN

v\_user := SYS\_CONTEXT('userenv', 'session\_user');

IF UPPER(v\_user) LIKE '%TEATRU\_CONSULTANT%' OR UPPER(v\_user) = 'TEATRU\_ADMIN' THEN

proc\_teatru\_ctx;

END IF;

END;

/

DECLARE

v\_user varchar2(30);

BEGIN

v\_user := SYS\_CONTEXT('userenv', 'session\_user');

IF UPPER(v\_user) LIKE '%TEATRU\_CONSULTANT%' OR UPPER(v\_user) = 'TEATRU\_ADMIN' THEN

proc\_teatru\_ctx;

END IF;

END;

/

După ce se creează triggerii *BEFORE UPDATE* și *BEFORE DELETE* de către admin, update-ul și insertul în tabelul reprezentații nu va mai putea fi realizat.

CREATE OR REPLACE TRIGGER trigger\_before\_update

BEFORE UPDATE ON reprezentatii

DECLARE

v\_ora NUMBER(3);

BEGIN

SELECT TO\_NUMBER(TO\_CHAR(sysdate, 'hh24')) INTO v\_ora

FROM dual;

IF SYS\_CONTEXT('teatru\_ctx', 'premis\_update') = 'nu' THEN

DBMS\_OUTPUT.PUT\_LINE(v\_ora || ' nu este permisa pentru update');

RAISE\_APPLICATION\_ERROR(-20301, 'Nu puteti face update la ora ' || v\_ora);

END IF;

END;

/

CREATE OR REPLACE TRIGGER trigger\_before\_delete

BEFORE DELETE ON reprezentatii

DECLARE

v\_ora NUMBER(3);

BEGIN

SELECT TO\_NUMBER(TO\_CHAR(sysdate, 'hh24')) INTO v\_ora

FROM dual;

IF SYS\_CONTEXT('teatru\_ctx', 'premis\_delete') = 'nu' THEN

DBMS\_OUTPUT.PUT\_LINE(v\_ora || ' nu este ora permisa pentru delete');

RAISE\_APPLICATION\_ERROR(-20300, 'Nu puteti face delete la ora ' || v\_ora);

END IF;

END;

/

SELECT TO\_NUMBER(TO\_CHAR(sysdate, 'hh24')) INTO v\_ora

FROM dual;

IF SYS\_CONTEXT('teatru\_ctx', 'premis\_delete') = 'nu' THEN

DBMS\_OUTPUT.PUT\_LINE(v\_ora || ' nu este ora permisa pentru delete');

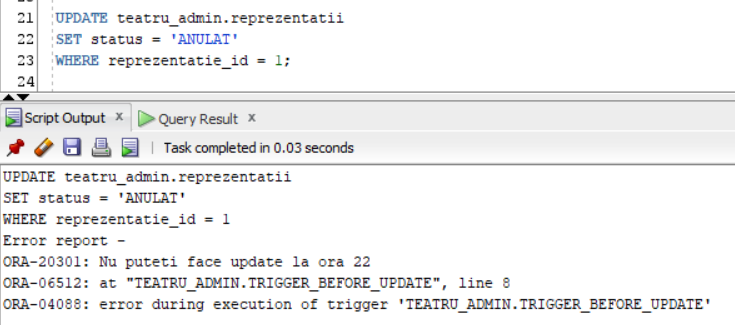
RAISE\_APPLICATION\_ERROR(-20300, 'Nu puteti face delete la ora ' || v\_ora);

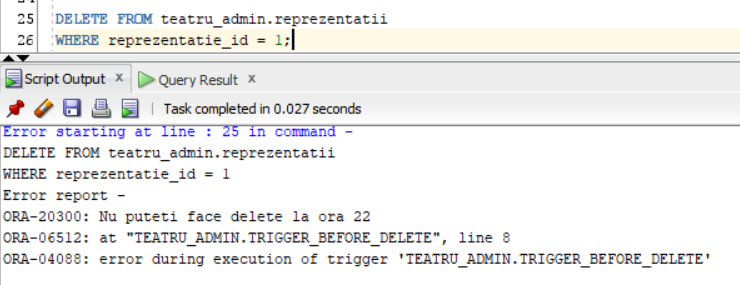
END IF;

END;

/

Rezultatul triggerului în intervalul nepotrivit:





6.2. SQL Injection

Construim următorul scenariu: presupunem că un utilizator de tipul SPECTATOR doreste sa isi modifice datele email și parola. Însă cele două câmpuri sunt criptate, iar în interfață nu vor fi vizibile la început. Pentru a le vedea, o fereastră de tip Pop-Up va aparea, în care utilizatorul trebuie să își introducă credențialele. Aici atacatorul va introduce date malițioase și va afla și datele decriptate (*email* și *telefon*) ale spectatorului.

CREATE OR REPLACE PROCEDURE get\_email\_and\_phone(v\_username VARCHAR2, v\_parola VARCHAR2)

AS

v\_id NUMBER(5) := -1;

v\_email RAW(100);

v\_phone RAW(100);

v\_decript\_email VARCHAR2(100);

v\_decript\_phone VARCHAR2(100);

BEGIN

EXECUTE IMMEDIATE 'SELECT spectator\_id FROM spectatori WHERE

username=''' || v\_username || ''' AND parola=hash\_parola(''' || v\_parola || ''')' INTO v\_id;

IF v\_id != -1 THEN

SELECT email

INTO v\_email

FROM spectatori

WHERE spectator\_id = v\_id;

SELECT numar\_telefon

INTO v\_phone

FROM spectatori

WHERE spectator\_id = v\_id;

v\_decript\_email := decriptare\_aes(v\_email, v\_id, 'spectatori');

v\_decript\_phone := decriptare\_aes(v\_phone, v\_id, 'spectatori');

DBMS\_OUTPUT.PUT\_LINE('Email-ul este: ' || v\_decript\_email || ' iar telefonul este: ' || v\_decript\_phone);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare');

END;

/

DBMS\_OUTPUT.PUT\_LINE('Email-ul este: ' || v\_decript\_email || ' iar telefonul este: ' || v\_decript\_phone);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

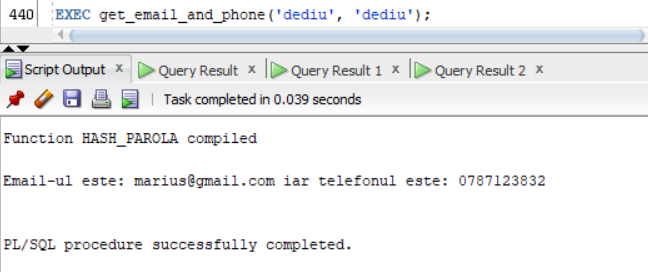
DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

WHEN OTHERS THEN

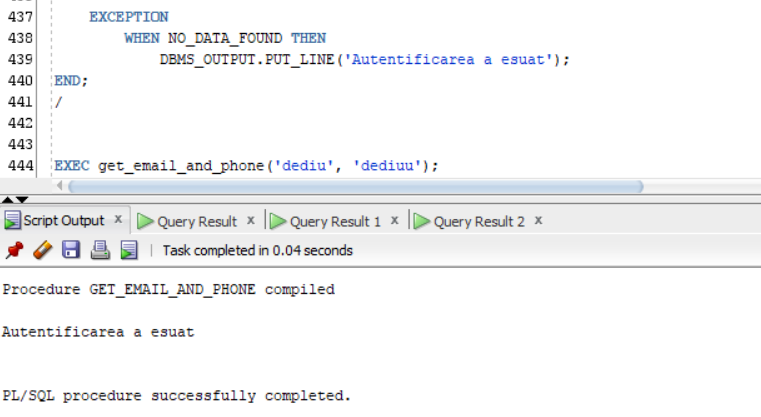
DBMS\_OUTPUT.PUT\_LINE('Eroare');

END;

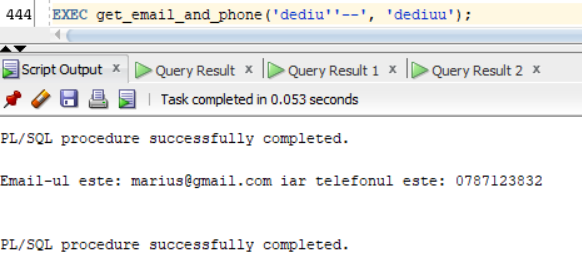
/

 *Test 1*: Nume și parolă corecte, autentificare cu succes

*Test 2*: Nume corect și parolă greșită, autentificare eșuată



*Test 3:* Atac malițios – se cunoaște un username și fără parolă (*'dediu''--'*), atacatorul va afla email-ul și telefonul



Vom înlocui concatenarea șirului cu *bind\_variables,* iar în acest mod nu vom mai fi vulnerabili la atac.

CREATE OR REPLACE PROCEDURE get\_email\_and\_phone\_safe(v\_username VARCHAR2, v\_parola VARCHAR2)

AS

v\_id NUMBER(5) := -1;

v\_email RAW(100);

v\_phone RAW(100);

v\_decript\_email VARCHAR2(100);

v\_decript\_phone VARCHAR2(100);

error\_msg VARCHAR2(200);

BEGIN

EXECUTE IMMEDIATE 'SELECT spectator\_id FROM spectatori WHERE

username=:nume AND parola=hash\_parola(:pass)' INTO v\_id

USING v\_username, v\_parola;

IF v\_id != -1 THEN

SELECT email

INTO v\_email

FROM spectatori

WHERE spectator\_id = v\_id;

SELECT numar\_telefon

INTO v\_phone

FROM spectatori

WHERE spectator\_id = v\_id;

v\_decript\_email := decriptare\_aes(v\_email, v\_id, 'spectatori');

v\_decript\_phone := decriptare\_aes(v\_phone, v\_id, 'spectatori');

DBMS\_OUTPUT.PUT\_LINE('Email-ul este: ' || v\_decript\_email || ' iar telefonul este: ' || v\_decript\_phone);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

WHEN OTHERS THEN

error\_msg := SUBSTR(SQLERRM,1,100);

DBMS\_OUTPUT.PUT\_LINE(error\_msg);

END;

/

SELECT numar\_telefon

INTO v\_phone

FROM spectatori

WHERE spectator\_id = v\_id;

v\_decript\_email := decriptare\_aes(v\_email, v\_id, 'spectatori');

v\_decript\_phone := decriptare\_aes(v\_phone, v\_id, 'spectatori');

DBMS\_OUTPUT.PUT\_LINE('Email-ul este: ' || v\_decript\_email || ' iar telefonul este: ' || v\_decript\_phone);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Autentificarea a esuat');

WHEN OTHERS THEN

error\_msg := SUBSTR(SQLERRM,1,100);

DBMS\_OUTPUT.PUT\_LINE(error\_msg);

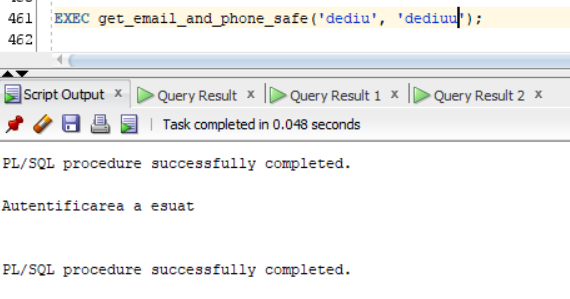
END;

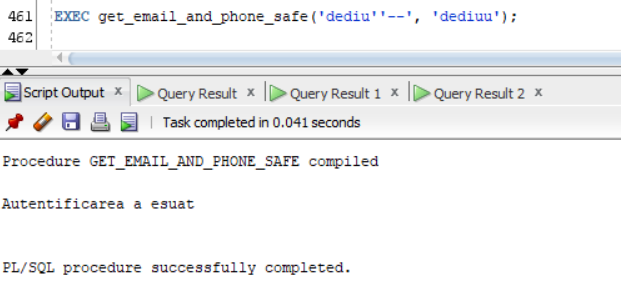
/

*Test 1*: Nume și parolă corecte, autentificare cu succes



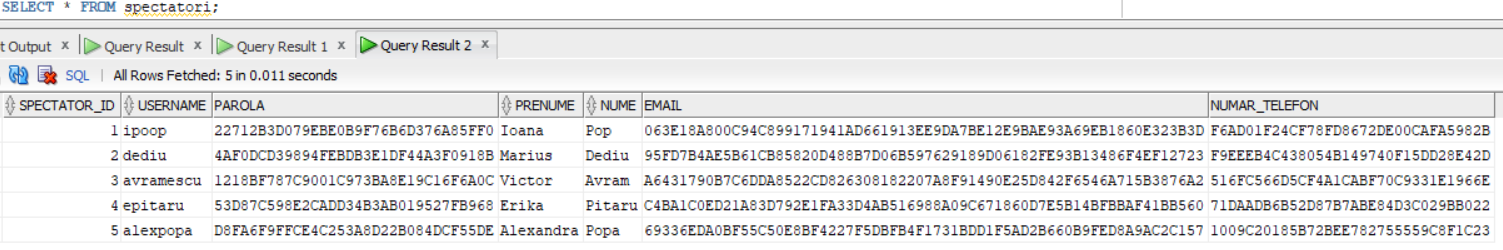
*Test 2*: Nume corect și parolă greșită, autentificare eșuată

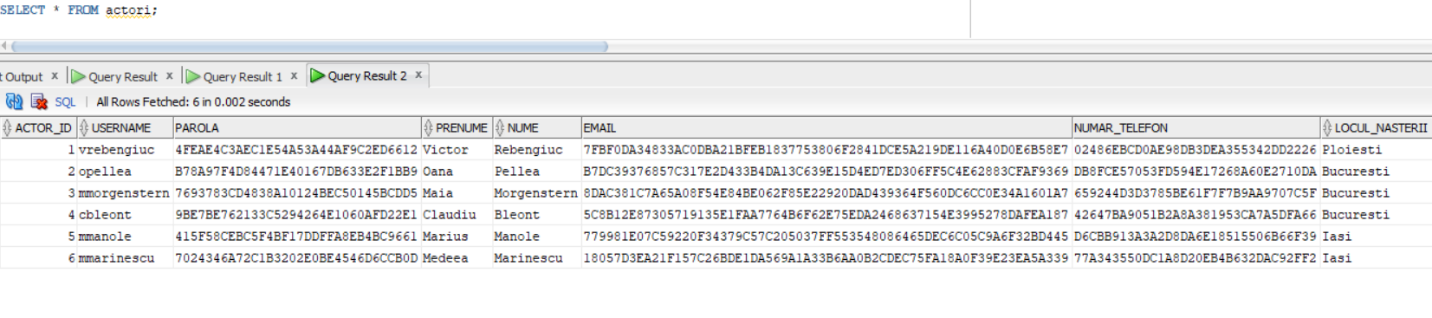
**

*Test 3:* Încercare de atac malițios – *CU EȘEC*

1. Mascarea datelor

Vom masca datele din cadrul tabelelor SPECTATORI și ACTORI.

Datele din tabelul *SPECTATORI*:

Datele din tabelul *ACTORI*:

Pentru tabelul SPECTATORI, coloanele care vor fi mascate sunt: *spectator\_id,* username, *prenume* și *nume*, întrucât celelalte date sunt deja criptate. Cum coloana *spectator\_id* este cheie externă atât în tabelul BILETE, cât și în tabelul RECENZII, trebuie să mascăm și id-urile corespunzătoare din tabelele referite.

În cazul tabelului ACTORI vom masca numai coloana *username*, deoarece restul datelor nu sunt confidențiale, fiind referite persoane publice.

CREATE OR REPLACE PACKAGE pachet\_mascare IS

FUNCTION f\_mascare(sir varchar2) RETURN varchar2;

FUNCTION f\_mascare(nr number) RETURN number;

END;

/

CREATE OR REPLACE PACKAGE BODY pachet\_mascare IS

TYPE tip\_tabind IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

v\_tabind tip\_tabind;

FUNCTION f\_mascare(sir varchar2) RETURN varchar2

IS

v\_sir varchar2(100);

v\_lung number;

BEGIN

v\_sir:=substr(sir,1,1);

SELECT length(sir) INTO v\_lung

FROM dual;

v\_sir:=rpad(v\_sir,v\_lung,'\*'); --pastreaza prima litera si adauga stelute in locul restului literelor

RETURN v\_sir;

END f\_mascare;

v\_lung number;

BEGIN

v\_sir:=substr(sir,1,1);

SELECT length(sir) INTO v\_lung

FROM dual;

v\_sir:=rpad(v\_sir,v\_lung,'\*'); --pastreaza prima litera si adauga stelute in locul restului literelor

RETURN v\_sir;

END f\_mascare;

FUNCTION f\_mascare(nr number) RETURN number

IS

lung number;

minnou number;

maxnou number;

l\_seed VARCHAR2(100);

v\_nrnou number;

BEGIN

IF v\_tabind.EXISTS(nr) THEN

RETURN v\_tabind(nr); -- pentru cheia externă

ELSE -- generam numere de lungimea nr initial + 1, care sa inceapa cu prima cifra

lung := LENGTH(to\_char(nr)) + 1;

minnou := to\_number(rpad(substr(to\_char(nr),1,1),lung,'0'));

maxnou := to\_number(rpad(substr(to\_char(nr),1,1),lung,'9'));

dbms\_output.put\_line('minnou='||minnou||' maxnou='||maxnou);

l\_seed := TO\_CHAR(SYSTIMESTAMP,'YYYYDDMMHH24MISSFFFF');

DBMS\_RANDOM.seed (val => l\_seed);

v\_nrnou := round(DBMS\_RANDOM.value(low=>minnou,high=> maxnou),0);

v\_tabind(nr) := v\_nrnou;

RETURN v\_nrnou;

END IF;

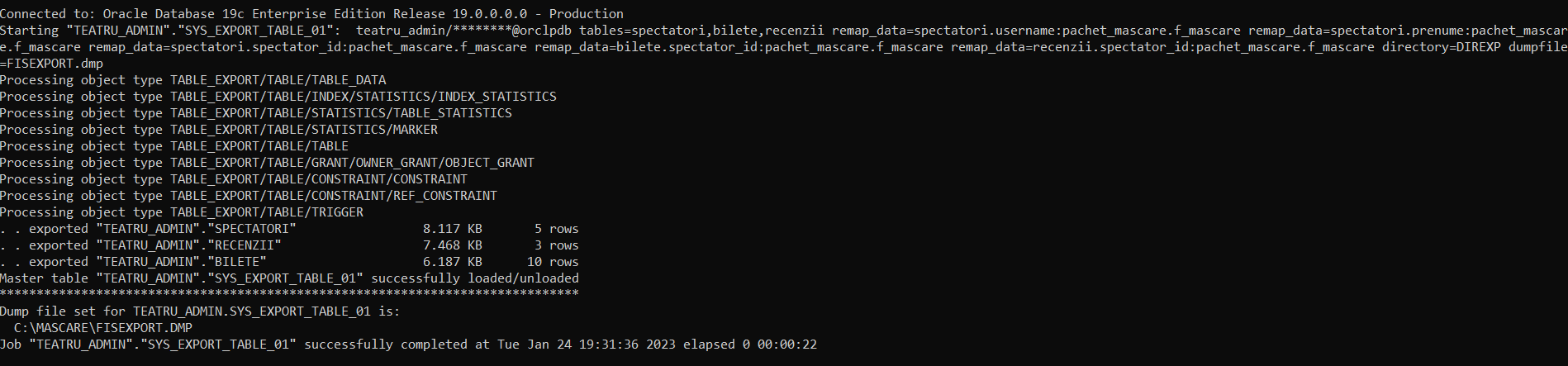
END f\_mascare;

END; /

Comandă mascare date:

expdp teatru\_admin/admin1@orclpdb tables=spectatori,bilete,recenzii,actori remap\_data=spectatori.username:pachet\_mascare.f\_mascare remap\_data=spectatori.prenume:pachet\_mascare.f\_mascare remap\_data=spectatori.nume:pachet\_mascare.f\_mascare remap\_data=spectatori.spectator\_id:pachet\_mascare.f\_mascare remap\_data=bilete.spectator\_id:pachet\_mascare.f\_mascare remap\_data=recenzii.spectator\_id:pachet\_mascare.f\_mascare remap\_data=actori.username:pachet\_mascare.f\_mascare directory=DIREXP dumpfile=FISEXPORT.dmp

Rezultat:

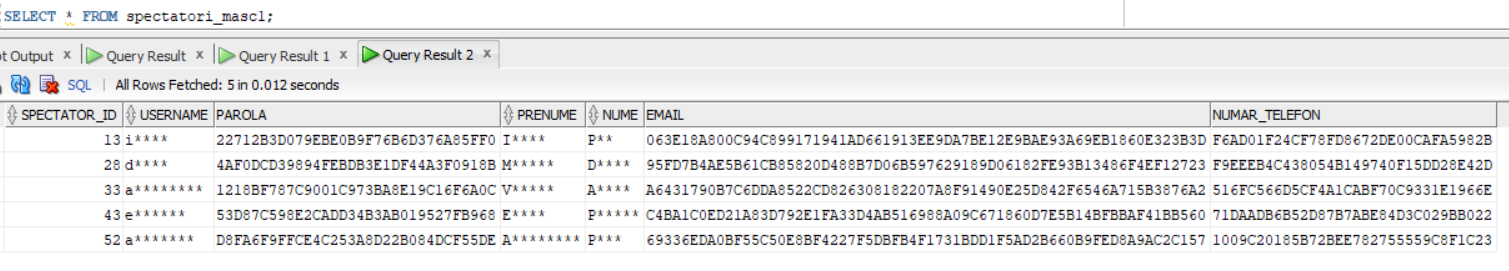


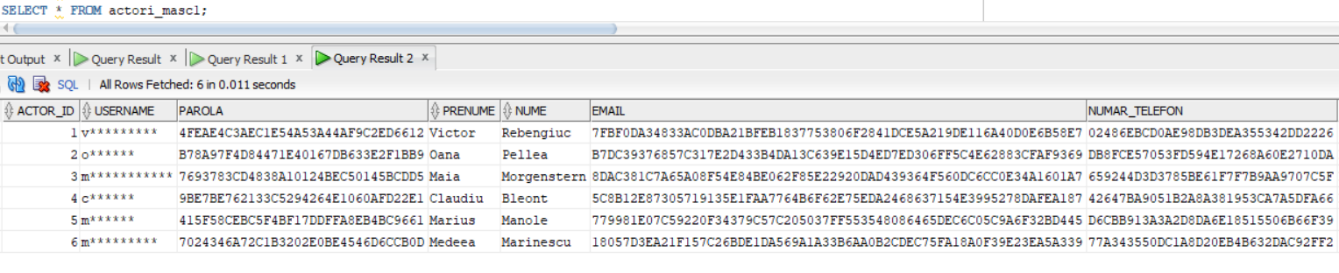
Comandă pentru exportarea tabelelor:

impdp teatru\_admin/admin1@orclpdb directory=DIREXP dumpfile=FISEXPORT.DMP TABLES=spectatori,bilete,recenzii,actori remap\_table=spectatori:spectatori\_masc1 remap\_table=bilete:bilete\_masc1 remap\_table=recenzii:recenzii\_masc1 remap\_table=actori:actori\_masc1

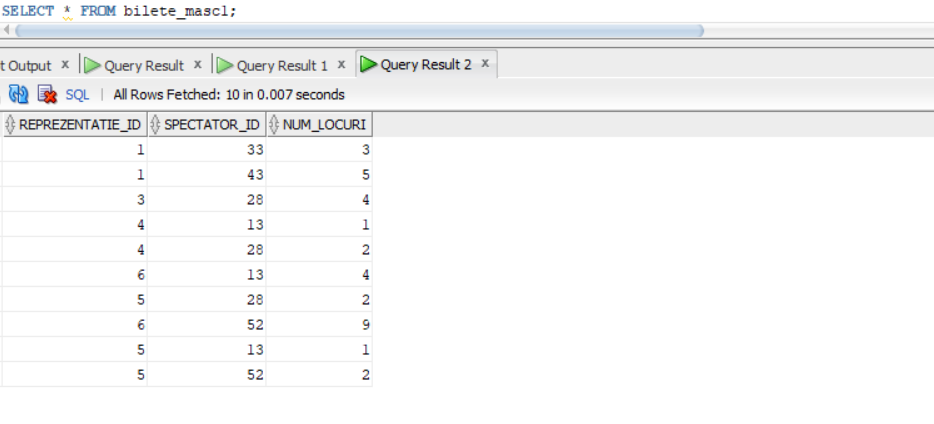
Rezultate în urma mascării:

*SPECTATORI\_MASC1*



*ACTORI\_MASC1*

*BILETE\_MASC1*



*RECENZII\_MASC1*

