Gestionarea datelor unor biblioteci

**1. Prezentați pe scurt baza de date (utilitatea ei).**

În acest proiect, modelul de date va gestiona informați legate de organizarea si funcționarea unor biblioteci. Fiecare bibliotecă oferă accesul la cărțile ei, prin intermediul împrumutului, fiecărui membru al ei. Membri unei biblioteci sunt nevoiți sa dețină un permis de intrare pentru fiecare bibliotecă la care sunt înscriși.

Fiecare bibliotecă poate face o comandă, de la unul din furnizori care lucrează cu o editură. O editură poate lucra cu mai mulți furnizori.

Cărțile sunt catalogate după tipuri (ex: Cărți de poezii, Cărți pentru copii, Cărți științifice, etc), iar fiecare dintre aceste cărți se impart în subtipuri (ex: Cărți științifice despre biologie, matmatică, informatică etc).

Acest model este util deoarece ține evidența cărților împrumutate, comenzilor, membrilor și catalogarea cărților.

Modelul de date respectă anumite restricți de funcționare:

- Fiecare carte trebuie să aibă o editură.

- O carte are un singur autor.

- Un membru al biblioteci are un singur permis de intrare pentru o bibliotecă.

- Un membru nu poate împrumuta o carte fără permisul de intrare specific biblioteci din care dorește să împrumute.

- Un membru nu poate împrumuta aceeași carte de mai multe ori în aceeași zi.

- Un furnizor și o editură se află la o singură locație.

- Un angajat poate sa conducă o singură bibliotecă.

- Toate adresele se află în România.

- O carte poate avea un tip și mai multe subtipuri.

- Presupunem că nu există autori cu același tuplu de nume și prenume

**Entități**

Pentru modelul de date referitor la gestiunea unor biblioteci, structurile ANGAJATI, BIBLIOTECI, FUNCTII, SUBTIPURI, AUTORI, CARTI, ADRESE, PERMIS\_DE\_INTRARE, COMENZI, FURNIZORI, EDITURI, CITITORI reprezintă entități.

ANGAJATI = persoane care se ocupă cu funcționarea propiu-zisă a unei biblioteci. Cheia primară este *angajat\_id.*

FUNCTII = funcțiile pe care le pot avea angajații unei biblioteci. Cheia primară este *functie\_id.*

BIBLIOTECI = clădirea special amenajată pentru stocarea cărților. Aceasta poate oferi posibilitatea membrilor săi de a împrumuta cărțile stocate in ea. Cheia primară este *biblioteca\_id.*

ADRESE = date care indică localizarea pe hartă a unei persoane sau instituții. Pentru acest model vom considera că toate adresele se află în România. Cheia primară este *adresa\_id.*

AUTORI = persoană care a scris o carte. O carte poate avea un singur autor. Cheia primară este *autor\_id.*

CARTI = scriere cu un anumit subiect, tipărită și legată sau broșată în volum. Cheia primară este *carte\_id.*

CITITORI = persoană care citește o carte. Cheia primară este *cititor\_id.*

PERMIS\_DE\_INTRARE = document cu scopul de a permite accesul la cărțile aflate într-o bibliotecă. Cheia primară este *permis\_id.*

EDITURI = este o societate comercială care își joacă rolul pe piața cărților. Cheia primară este *editura\_id.*

FURNIZORI = companie care se ocupa cu distribuirea și producerea unor bunuri. Cheia primară este *furnizor\_id.*

COMENZI = acțiunea de a cumpăra produse de la un furnizor. Cheia primară este *comanda\_is.*

SUBTIPURI = subtipul unei cărți. Cheia primară este *subtip\_id.*

**Relații**

ANGAJATI\_are\_FUNCTII = relația care leagă entitățile ANGAJATI și FUNCTII reflectând legătura dintre acestea (ce funcție are un angajat). Aceasta are cardinalitatea maxima 1:n (un angajat poate avea o singură funcție și mai mulți angajați pot avea aceeași funcție) și cardinalitatea minimă este 0:1(Un angajat trebuie să aibă o functie și un angajat nu trebuie să aibă o anume funcție).

ANGAJATI\_conduce\_BIBLIOTECI = relația care leagă entitățile ANGAJAT și BIBLIOTECI refletând legătura dintre acestea (ce angajat conduce o bibliotecă). Relația maximă este 1:1 (un angajat poate să conducă o bibliotecă si o bibliotecă poate fi condusă de un angajat) și relația minimă 0:0 (un angajat nu trebuie să conducă o bibliotecă și o bibliotecă nu trebuie să fie condusă de un angajat).

ANGAJATI\_lucreaza\_BIBLIOTECI = relația care leagă entitățile ANGAJATI și BIBLIOTECI refletând legătura dintre acestea (ce angajați lucrează într-o bibliotecă). Relația maximă este 1:n (un angajat poate lucra într-o bibliotecă și într-o bibliotecă pot lucra mai mulți angajați) și relația minimă 0:1 (un angajat trebuie să lucreze într-o bibliotecă și într-o bibliotecă nu trebuie să lucreze un angajat).

ANGAJATI\_locuiesc\_ADRESE = relația care leagă entitățile ANGAJATI și ADRESE reflectând legătura dintre acestea (unde locuiesc angajați). Relația maximă este m:n (angajați pot locui la mai multe adrese și la o adresă pot locui mai mulți angajați) și relația minimă este 0:1(un angajat trebuie sa locuiască la o adresă și la o adresă nu trebuie să locuiască un angajat).

PERMIS\_DE\_INTRARE\_apartine\_BIBLIOTECI = relația care leagă entitățile BIBLIOTECI și PERMIS\_DE\_INTRARE reflectând legătura dintre acestea (de ce bibliotecă aparține permisul de intrare). Relația maximă este 1:n (un permis de intrare poate aparține de o singură bibliotecă și de o bibliotecă pot aparține mai multe permisuri de intrare) și relația minimă este 0:1 (un permis de intrare trebuie să aparțina de o bibliotecă și de o bibliotecă nu trebuie să aparțină un permis de intrare).

BIBLIOTECI\_se\_afla\_ADRESE = relația care leagă entitățile BIBLIOTECI și ADRESE reflectând legătura dintre acestea (unde se află o bibliotecă). Relația maximă este 1:n (mai multe biblioteci se pot afla la o adresă și o bibliotecă se poate afla la o adresă) și relația minimă 0:1 (o bibliotecă trebuie să se afle la o adresă și la o adresă nu trebuie să se afle o bibliotecă).

BIBLIOTECI\_face\_COMENZI = relația care leagă entitățile BIBLIOTECI și COMENZI reflectănd comenzile făcute de o bibliotecă. Relația maximă este 1:n (o bibliotecă poate face mai multe comenzi și o comandă poate să fie făcută de o bibliotecă) și relația minimă este 0:1 (o bibliotecă nu trebuie să facă o comandă și o comandă trebuie să fie făcută de o bibliotecă).

PERMIS\_DE\_INTRARE\_apartine\_CITITORI = relația de tip 3 care leagă entitățile PERMIS\_DE\_INTRARE și CITIORI reflectând permisurile de intrare ale cititorilor. Relația maximă este 1:n (un permis poate aparține unui cititor și un cititor poate avea mai multe permisuri) și relația minimă este 0:1 (un permis de intrare trebuie să aparțină unui cititor și un cititor nu trebuie să aibă un anume permis).

CITITORI\_imprumuta\_CARTI\_de\_la\_BIBLIOTECI\_cu\_un\_PERMIS\_DE\_INTRARE = relație de tip 3 care leagă entitățile PERMIS\_DE\_INTRARE, CITITORI, BIBLIOTECI și CARTI reflectând cărțile împrumutate de un cititor de la o biblioteca folosind un permis.

CITITORI\_locuiesc\_ADRESE = relație care leagă entitățile CITITORI și ADRESE reflectănd adresele cititorilor. Relația maximă este m:n (un cititor se poate afla la mai multe adrese și la o adresă se pot afla mai mulți cititori) și relația minimă 0:1 (la o adresă nu trebuie să locuiască un cititor și un cititor trebuie să locuiască la o adresă).

EDITURI\_se\_afla\_ADRESE = relație care leagă entitățile EDITURI și ADRESE reflectănd locația unde se află o editură. Relația maximă este 1:1 (o editura se poate afla la o adresă și la o adresă se poate afla o editură) și relația minimă 0:1 (o editură trebuie să se afle la o adresă și la o adresă nu trebuie să se afle o editură).

FURNIZORI\_se\_afla\_ADRESE = relație care leagă entitățile FURNIZORI și ADRESE reflectănd locația unde se află un furnizor. Relația maximă este 1:1 (un furnizor se poate afla la o adresă și la o adresă se poate afla un furnizor) și relația minimă 0:1 (un furnizor trebuie să se afle la o adresă și la o adresă nu trebuie să se afle un furnizor).

EDITURI\_lucreaza\_cu\_FURNIZORI = relația care leagă entitățile EDITURI și FURNIZORI reflectănd legătura dintre acestea (ce edituri lucreaza cu ce furnizori). Relația maximă este m:n (o editură poate lucra cu mai mulți furnizori și un furnizor poate lucra cu mai multe edituri) și relația minimă este 0:1 (o editură trebuie să lucreze cu un furnizor și un furnizor nu trebuie să lucreze cu o editură).

FURNIZORI\_se\_afla\_COMENZI = relația care leagă entitățile FURNIZORI și COMENZI reflectănd legătura dintre acestea (în ce comenzi se află un furnizor). Relația maximă este 1:n (un furnizor se poate afla în mai multe comenzi și într-o comandă se poate afla un furnizor) și relația minimă 0:1 (într-o comandă trebuie să se afle un furnizor și un furnizor nu trebuie să se afle într-o comandă).

CARTI\_se\_afla\_COMENZI = relația care leagă entitățile CARTI și FURNIZORI reflectând legătura dintre acestea (în ce comenzi se află o carte). Relația maximă este 1:n (într-o comandă se poate afla o carte și o carte se poate afla în mai multe comenzi) și relația minimă este 0:1 (într-o comandă trebuie să se afle o carte și o carte nu trebuie să se afle într-o comanda).

CARTI\_are\_EDITURI = relația care leagă entitățile CARTI și EDITURI reflectând legătura dintre acestea (ce editură are o carte). Relația maximă este 1:n (o carte poate avea o editură și o editură poate avea mai multe cărți) și relația minimă 0:1 (o carte trebuie să aibă o editură și o editură nu trebuie să aibă o carte).

AUTORI\_scris\_CARTI = relația care leagă CARTI și AUTORI reflectând legătura dintre acestea (ce autor a scris o carte). Relația maximă este 1:n (o carte poate fi scrisă de un autor și un autor poate scrie mai multe cărți) și relația minimă este 0:1 (o carte trebuie să fie scrisă de un autor și un autor nu trebuie să scrie o carte).

CARTI\_are\_SUBTIPURI = relația care leagă entitățile CARTI și SUBTIPURI reflectând legătura dintre acestea (ce subtipuri are o carte). Relația maximă este m:n (o carte poate avea mai multe subtipuri și mai multe cărți pot avea un subtip) și relația minimă 0:1 (o carte trebuie să aibă un subtip și o carte nu trebuie să aibă un anume subtip).

**Atribute**

Atributele entități ANGAJATI sunt: **angajat\_id** (variabilă de tip întreg, de lungime maximă 5 și este cheia primară a angajatului), **nume** (variabilă de tip caracter, de lungime maximă 25 și nu este nulă), **prenume** (variabilă de tip caracter, de lungime maximă 25 și nu este nulă), **cnp** (variabilă de tip caracter, de lungime maximă 13 și nu este nulă), **telefon** (variabilă de tip caracter, de lungime maximă 20 și nu este nulă), **email** (variabilă de tip caracter, de lungime maximă 320, și nu este nulă), **salariu** (variabilă de tip număr real, de lungime maximă 10 și nu este nulă), **data\_angajare** (variabilă de tip dată, care reprezintă data angajări angajatului, implicit este egală cu data curentă din sysdate), **functie\_id** (variabilă de tip întreg, de lungime maximă 5, nu este nulă, este cheia externă ce face legătura cu entitatea FUNCTII și reprezintă funcția pe care o are angajatul în bibliotecă), **biblioteca\_id** (variabilă de tip întreg, de lungime maximă 5, nu este nulă, este cheie externă ce face legătură cu entitatea BIBLIOTECI și reprezintă biblioteca în care lucrează angajatul).

Atributele entități FUNCTII sunt: **functie\_id** (variabilă de tip întreg, de lungime maximă 5 și este cheia primară a funcției), **denumire** (variabilă de tip caracter, de lungime maximă 30 și nu este nulă), **salariu\_minim** (variabilă de tip număr real, de lungime maximă 10, nu este nulă și este mai mic sau egal cu salariul maxim), **salariu\_maxim** (variabilă de tip număr real, de lungime maximă 10, nu este nulă și este mai mare sau egal cu salariul minim).

Atributele entități BIBLIOTECI sunt: **biblioteca\_id** (variabilă de tip întreg, de lungime maxima 5 și este cheia primară a biblioteci), **denumire** (variabilă de tip caracter de lungime maximă 30 și nu este nulă), **telefon** (variabilă de tip caracter, de lungime maximă 20, nu este nulă și reprezintă numărul de telefon la care poate fi contactată o bibliotecă), **manager\_id** (variabilă de tip întreg, de dimensiune maximă 5, este cheie externă ce face legătura cu entitatea ANGAJATI și reprezintă angajatul care se ocupă de condusul biblioteci), **adresa\_id** (este o variabilă de tip întreg, de lungime maximă 5, nu este nulă, este o cheie externă ce face legătura cu entitatea ADRESE și reprezintă adresa la care se află biblioteca).

Atributele entități ADRESE sunt: **adresa\_id** (variabilă de tip întreg, de lungime maximă 5 și este cheia primară a adresei), **judet** (variabilă de tip caracter, de lungime maximă 50 și nu este nulă), **localitate** (variabilă de tip caracter, de lungime maximă 50 și nu este nulă), **cod\_postal** (variabilă de tip caracter, de lungime maximă 6 și nu este nulă), **strada** (variabilă de tip caracter, de lungime maximă 50).

Atribute entități AUTORI sunt: **autor\_id** (variabilă de tip întreg, de lungime maximă 5 și este cheia primară a autorului), **nume** (variabilă de tip caracter, de lungime maximă 25 și nu este nulă), **prenume** (variabilă de tip caracter, de lungime maximă 25 și nu este nulă).

Atributele entități CARTI sunt: **carte\_id** (variabilă de tip întreg, de lungime maximă 5 și este cheia primară a cărți), **titlu** (variabilă de tip caracter, de lungime maximă 50 și nu este nulă), **editura\_id** (variabilă de tip întreg, de dimensiune maximă 5, nu este nulă, este cheie externă ce face legătura cu entitatea EDITURI și reprezintă editura cărți), **an\_aparitie** (variabilă de tip întreg, de lungime maximă 5 care reprezintă anul apariției cărți și nu este nulă), **autor\_id** (variabilă de tip întreg, de dimensiune maximă 5, nu este nulă, este cheie externă ce face legătura cu entitatea AUTORI și reprezintă autorul cărți), **tip\_carte** (variabilă de tip caracter, de lungime maximă 25 și nu este nulă).

Atributele entități CITITORI sunt: **cititor\_id** (variabilă de tip întreg, de lungime maximă 5, și este cheia primară a cititorului), **nume** (variabilă de tip caracter, de lungime maximă 25 și nu este nulă), **prenume** (variabilă de tip caracter, de lungime maximă 25 și nu este nulă), **cnp** (variabilă de tip caracter, de lungime maximă 13 și nu este nulă), **telefon** (variabilă de tip caracter, de lungime maximă 20 și nu este nulă), **email** (variabilă de tip caracter, de lungime maximă 320, și nu este nulă).

Atributele entități PERMIS\_DE\_INTRARE sunt: **permis\_id** (variabilă de tip întreg, de lungime maximă 5, și este cheia primară a permisului), **anul\_inscrieri** (variabilă de întreg, de lungime maximă 5 care reprezintă data înscrieri, implicit este egală cu data curentă din sysdate), **cititor\_id** (variabilă de tip întreg, de dimensiune maximă 5, nu este nulă, este cheie externă ce face legătura cu entitatea CITITORI și reprezintă deținătorul/deținătoarea permisului), **biblioteca\_id** (variabilă de tip întreg, de dimensiune maximă 5, nu este nulă, este cheie externă ce face legătura cu entitatea BIBLIOTECI și reprezintă biblioteca la care este valid permisul).

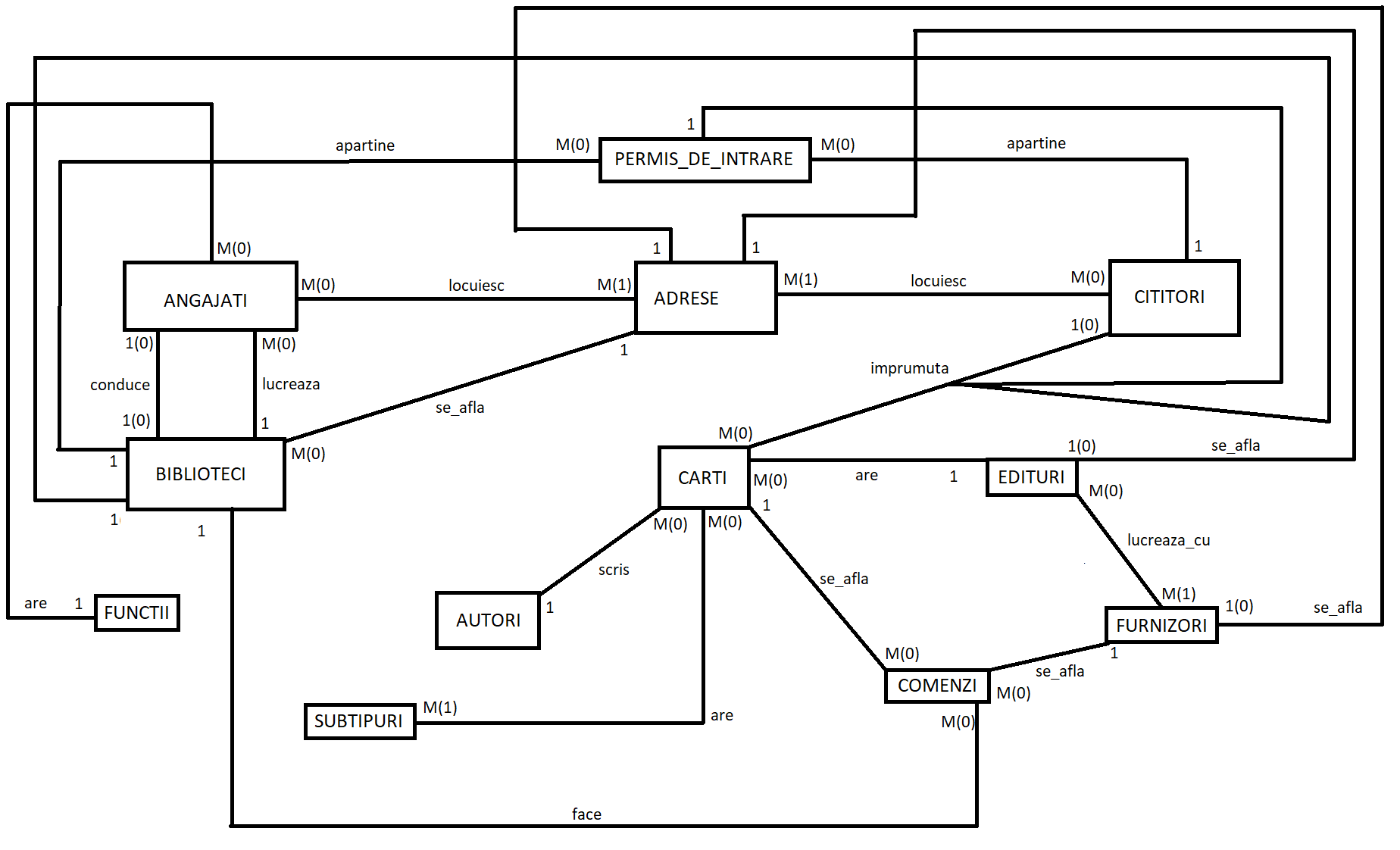
Atributele entități EDITURI sunt: **editura\_id** (variabilă de tip întreg, de lungime maximă 5, și este cheia primară a edituri), **denumire** (variabilă de tip caracter de lungime maximă 30 și nu este nulă), **telefon** (variabilă de tip caracter, de lungime maximă 20, nu este nulă și reprezintă numărul de telefon la care poate fi contactată o editură), **email** (variabilă de tip caracter, de lungime maximă 320, și nu este nulă), **adresa\_id** (este o variabilă de tip întreg, de lungime maximă 5, nu este nulă, este o cheie externă ce face legătura cu entitatea ADRESE și reprezintă adresa la care se află editura).

Atributele entități FURNIZORI sunt: **furnizor\_id** (variabilă de tip întreg, de lungime maximă 5, și este cheia primară a edituri), **denumire** (variabilă de tip caracter de lungime maximă 30 și nu este nulă), **telefon** (variabilă de tip caracter, de lungime maximă 20, nu este nulă și reprezintă numărul de telefon la care poate fi contactat un furnizor), **email** (variabilă de tip caracter, de lungime maximă 320, și nu este nulă), **adresa\_id** (este o variabilă de tip întreg, de lungime maximă 5, nu este nulă, este o cheie externă ce face legătura cu entitatea ADRESE și reprezintă adresa la care se află furnizorul).

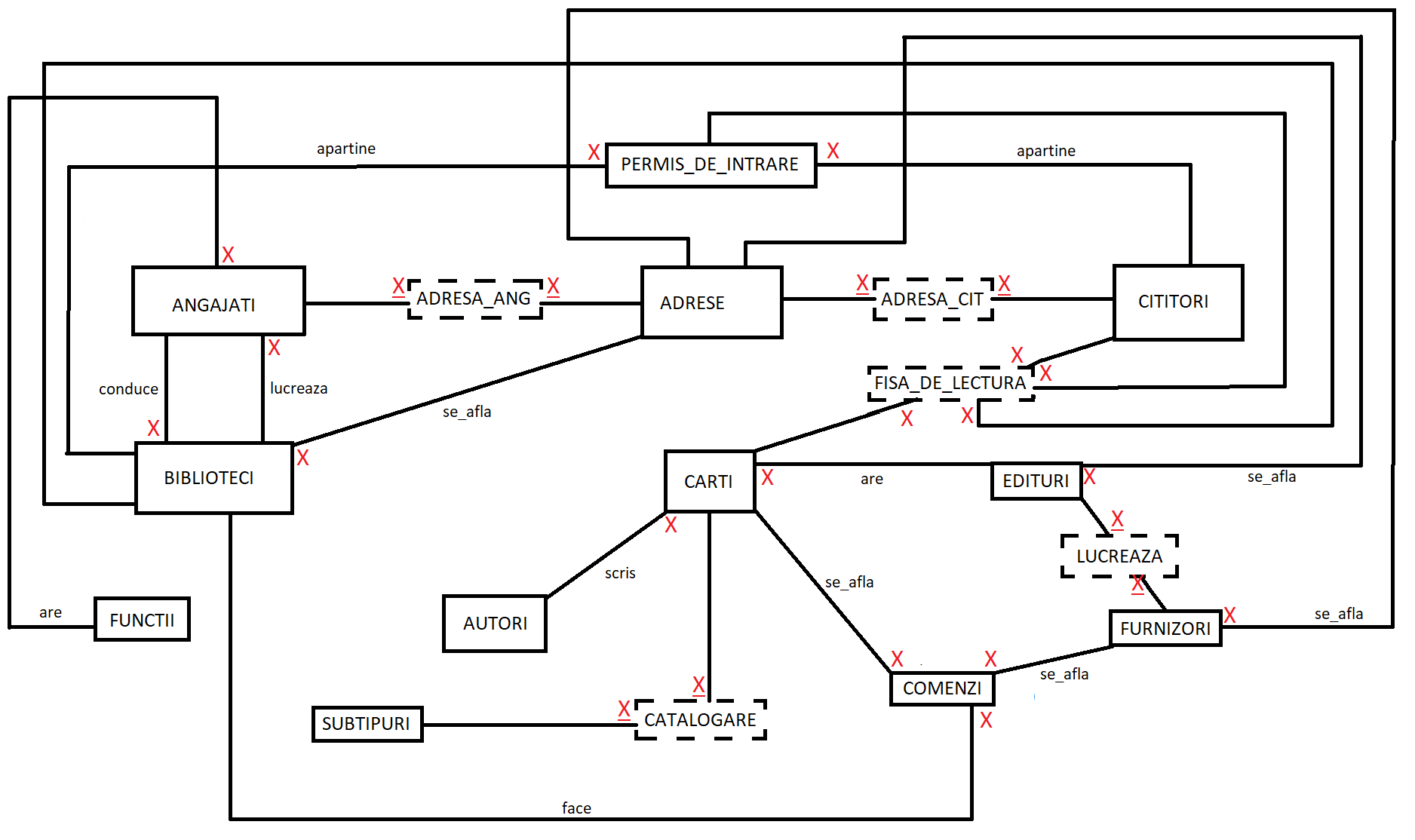
Atributele entități COMENZI sunt: **comanda\_id** (variabilă de tip întreg, de lungime maximă 5, și este cheia primară a comenzi), **furnizor\_id** (variabilă de tip întreg, de dimensiune maximă 5, este cheie externă ce face legătura cu entitatea FURNIZORI și reprezintă furnizorul care se ocupă de comandă), **carte\_id** (variabilă de tip întreg, de dimensiune maximă 5, este cheie externă ce face legătura cu entitatea CARTI și reprezintă cartea care este comandată), **biblioteca\_id** (variabilă de tip întreg, de dimensiune maximă 5, este cheie externă ce face legătura cu entitatea BIBLIOTECI și reprezintă biblioteca care a făcut comanda), **nr\_exemplare** (variabilă de tip întreg, de dimensiune maximă 10, nu este nulă și este mai mare ca 0), **pret\_comanda** (variabilă de tip număr real, de lungime maximă 10 și nu este nulă).

Atributele entități SUBTIPURI sunt: **subtip\_id** (variabilă de tip întreg, de lungime maximă 5, și este cheia primară a subtipului), **denumire** (variabilă de tip caracter de lungime maximă 30 și nu este nulă).

**2. Realizați diagrama entitate-relație (ERD).**

****

**3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.**



**Schemele relaționale** corespunzătoare diagramei conceptuale sunt:

ANGAJATI(angajat\_id#, nume, prenume, cnp, telefon, email, salariu, data\_angajare, functie\_id, biblioteca\_id)

FUNCTII(functie\_id#, denumire, salariu\_minim, salariu\_maxim)

BIBLIOTECI(biblioteca\_id#, denumire, telefon, manager\_id, adresa\_id)

ADRESE(adresa\_id#, judet, localitate, cod\_postal, strada)

AUTORI(autor\_id#, nume, prenume)

CARTI(carte\_id#, titlu, editura\_id, an\_aparitie, autor\_id, tip\_carte)

CITITORI(cititor\_id#, nume, prenume, cnp, telefon, email)

PERMIS\_DE\_INTRARE(permis\_id#, anul\_inscrieri, cititor\_id, biblioteca\_id)

EDITURI(editura\_id#, denumire, telefon, email, adresa\_id)

FURNIZORI(furnizor\_id#, denumire, telefon, email, adresa\_id)

COMENZI(comanda\_id#, furnizor\_id, carte\_id, biblioteca\_id, nr\_exemplare, pret\_comanda)

SUBTIPURI(subtip\_id#, denumire)

CATALOGARE(carte\_id#, subtip\_id#)

ADRESA\_ANG(angajat\_id#, adresa\_id#)

ADRESA\_CIT(cititor\_id#, adresa\_id#)

FISA\_DE\_LECTURA(fisa\_id#, cititor\_id, carte\_id, biblioteca\_id, permis\_id, data\_imprumut, data\_restituire)

LUCREAZA(editura\_id#, furnizor\_id#)

**4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).**

CREATE TABLE ADRESE(

adresa\_id NUMBER(5),

judet VARCHAR2(50) CONSTRAINT null\_judet\_adr not null,

localitate VARCHAR2(50) CONSTRAINT null\_localitate\_adr not null,

cod\_postal VARCHAR2(6) CONSTRAINT null\_cod\_postal not null,

strada VARCHAR2(50),

CONSTRAINT pk\_adr PRIMARY KEY(adresa\_id),

CONSTRAINT unq\_adresa\_adr UNIQUE(judet, localitate, cod\_postal, strada),

CONSTRAINT lungime\_cod\_postal\_adr CHECK(LENGTH(cod\_postal) = 6)

);

COMMIT;

CREATE TABLE FUNCTII(

functie\_id NUMBER(5),

denumire VARCHAR2(30) CONSTRAINT null\_denumire\_fun not null,

salariu\_minim NUMBER(10, 2) CONSTRAINT null\_salariu\_min\_fun not null,

salariu\_maxim NUMBER(10, 2) CONSTRAINT null\_salariu\_max\_fun not null,

CONSTRAINT pk\_fun PRIMARY KEY(functie\_id),

CONSTRAINT salariu\_min\_max\_logic CHECK(salariu\_minim <= salariu\_maxim),

CONSTRAINT unq\_denumire\_fun UNIQUE(denumire),

CONSTRAINT salariu\_min\_not\_negative\_fun CHECK(salariu\_minim > 0)

);

COMMIT;

CREATE TABLE ANGAJATI(

angajat\_id NUMBER(5),

nume VARCHAR2(25) CONSTRAINT null\_nume\_ang not null,

prenume VARCHAR2(25) CONSTRAINT null\_prenume\_ang not null,

cnp VARCHAR2(13) CONSTRAINT null\_cnp\_ang not null,

telefon VARCHAR2(20) CONSTRAINT null\_telefon\_ang not null,

email VARCHAR2(320) CONSTRAINT null\_email\_ang not null,

salariu NUMBER(10,2) CONSTRAINT null\_salariu\_ang not null,

data\_angajare DATE DEFAULT sysdate,

functie\_id NUMBER(5) CONSTRAINT null\_functie\_ang not null,

biblioteca\_id NUMBER(5) CONSTRAINT null\_biblioteca\_ang not null,

CONSTRAINT pk\_ang PRIMARY KEY(angajat\_id),

CONSTRAINT fk\_angajati\_functii\_ang FOREIGN KEY(functie\_id) REFERENCES FUNCTII(functie\_id) ON DELETE CASCADE,

CONSTRAINT unq\_cnp\_ang UNIQUE(cnp),

CONSTRAINT unq\_telefon\_ang UNIQUE(telefon),

CONSTRAINT unq\_email\_ang UNIQUE(email),

CONSTRAINT salariu\_logic\_ang CHECK(salariu > 0),

CONSTRAINT lungime\_telefon\_ang CHECK(LENGTH(telefon) >= 10)

);

COMMIT;

CREATE TABLE BIBLIOTECI(

biblioteca\_id NUMBER(5),

denumire VARCHAR2(30) CONSTRAINT null\_denumire\_bibl not null,

telefon VARCHAR2(20) CONSTRAINT null\_telefon\_bibl not null,

manager\_id NUMBER(5),

adresa\_id NUMBER(5) CONSTRAINT null\_adresa\_bibl not null,

CONSTRAINT pk\_bibl PRIMARY KEY(biblioteca\_id),

CONSTRAINT fk\_manager\_biblioteca\_bibl FOREIGN KEY(manager\_id) REFERENCES ANGAJATI(angajat\_id) ON DELETE SET NULL,

CONSTRAINT fk\_adresa\_biblioteca\_bibl FOREIGN KEY(adresa\_id) REFERENCES ADRESE(adresa\_id) ON DELETE CASCADE,

CONSTRAINT unq\_denumire\_bibl UNIQUE(denumire),

CONSTRAINT unq\_telefon\_bibl UNIQUE(telefon),

CONSTRAINT unq\_manager\_bibl UNIQUE(manager\_id),

CONSTRAINT lungime\_telefon\_bibl CHECK(LENGTH(telefon) >= 10)

);

ALTER TABLE ANGAJATI

ADD CONSTRAINT fk\_angajati\_biblioteci\_ang FOREIGN KEY(biblioteca\_id) REFERENCES BIBLIOTECI(biblioteca\_id) ON DELETE CASCADE;

COMMIT;

CREATE TABLE ADRESA\_ANG(

angajat\_id NUMBER(5),

adresa\_id NUMBER(5),

CONSTRAINT pk\_adrang PRIMARY KEY(angajat\_id, adresa\_id),

CONSTRAINT fk\_angajat\_adrang FOREIGN KEY(angajat\_id) REFERENCES ANGAJATI(angajat\_id) ON DELETE CASCADE,

CONSTRAINT fk\_adresa\_adrang FOREIGN KEY(adresa\_id) REFERENCES ADRESE(adresa\_id) ON DELETE CASCADE

);

COMMIT;

CREATE TABLE AUTORI(

autor\_id NUMBER(5),

nume VARCHAR2(25) CONSTRAINT null\_nume\_autor not null,

prenume VARCHAR2(25) CONSTRAINT null\_prenume\_autor not null,

CONSTRAINT pk\_aut PRIMARY KEY(autor\_id),

CONSTRAINT unq\_nume\_prenume\_autor UNIQUE(nume, prenume)

);

COMMIT;

CREATE TABLE EDITURI(

editura\_id NUMBER(5),

denumire VARCHAR2(30) CONSTRAINT null\_denumire\_edituri not null,

telefon VARCHAR2(20) CONSTRAINT null\_telefon\_edituri not null,

email VARCHAR2(320) CONSTRAINT null\_email\_edituri not null,

adresa\_id NUMBER(5) CONSTRAINT null\_adresa\_edituri not null,

CONSTRAINT pk\_edituri PRIMARY KEY(editura\_id),

CONSTRAINT fk\_adresa\_edituri FOREIGN KEY(adresa\_id) REFERENCES ADRESE(adresa\_id) ON DELETE CASCADE,

CONSTRAINT unq\_denumire\_edituri UNIQUE(denumire),

CONSTRAINT unq\_telefon\_edituri UNIQUE(telefon),

CONSTRAINT unq\_email\_edituri UNIQUE(email),

CONSTRAINT lungime\_telefon\_edituri CHECK(LENGTH(telefon) >= 10)

);

COMMIT;

CREATE TABLE CARTI(

carte\_id NUMBER(5),

titlu VARCHAR2(50) CONSTRAINT null\_titlu\_car not null,

editura\_id NUMBER(5) CONSTRAINT null\_editura\_car not null,

an\_aparitie NUMBER(5) CONSTRAINT null\_an\_car not null,

autor\_id NUMBER(5) CONSTRAINT null\_autor\_car not null,

tip\_carte VARCHAR2(25) CONSTRAINT null\_tip\_car not null,

CONSTRAINT pk\_car PRIMARY KEY(carte\_id),

CONSTRAINT fk\_autor\_carte\_car FOREIGN KEY(autor\_id) REFERENCES AUTORI(autor\_id) ON DELETE CASCADE,

CONSTRAINT fk\_editura\_carte\_car FOREIGN KEY(editura\_id) REFERENCES EDITURI(editura\_id) ON DELETE CASCADE

);

COMMIT;

CREATE TABLE CITITORI(

cititor\_id NUMBER(5),

nume VARCHAR2(25) CONSTRAINT null\_nume\_cit not null,

prenume VARCHAR2(25) CONSTRAINT null\_prenume\_cit not null,

cnp VARCHAR2(13) CONSTRAINT null\_cnp\_cit not null,

telefon VARCHAR2(20) CONSTRAINT null\_telefon\_cit not null,

email VARCHAR2(320) CONSTRAINT null\_email\_cit not null,

CONSTRAINT pk\_cit PRIMARY KEY(cititor\_id),

CONSTRAINT unq\_cnp\_cit UNIQUE(cnp),

CONSTRAINT unq\_telefon\_cit UNIQUE(telefon),

CONSTRAINT unq\_email\_cit UNIQUE(email),

CONSTRAINT lungime\_telefon\_cit CHECK(LENGTH(telefon) >= 10)

);

COMMIT;

CREATE TABLE ADRESA\_CIT(

cititor\_id NUMBER(5),

adresa\_id NUMBER(5),

CONSTRAINT pk\_adrcit PRIMARY KEY(cititor\_id, adresa\_id),

CONSTRAINT fk\_cititor\_adrcit FOREIGN KEY(cititor\_id) REFERENCES CITITORI(cititor\_id) ON DELETE CASCADE,

CONSTRAINT fk\_adresa\_adrcit FOREIGN KEY(adresa\_id) REFERENCES ADRESE(adresa\_id) ON DELETE CASCADE

);

COMMIT;

CREATE TABLE PERMIS\_DE\_INTRARE(

permis\_id NUMBER(5),

anul\_inscrieri NUMBER(5) DEFAULT TO\_CHAR(sysdate, 'YYYY'),

cititor\_id NUMBER(5) CONSTRAINT null\_cititor\_permis not null,

biblioteca\_id NUMBER(5) CONSTRAINT null\_biblioteca\_permis not null,

CONSTRAINT pk\_permis PRIMARY KEY(permis\_id),

CONSTRAINT fk\_cititor\_permis FOREIGN KEY(cititor\_id) REFERENCES CITITORI(cititor\_id) ON DELETE CASCADE,

CONSTRAINT fk\_biblioteca\_permis FOREIGN KEY(biblioteca\_id) REFERENCES BIBLIOTECI(biblioteca\_id) ON DELETE CASCADE,

CONSTRAINT unq\_biblioteca\_cititor\_permis UNIQUE(cititor\_id, biblioteca\_id)

);

COMMIT;

CREATE TABLE FISA\_DE\_LECTURA(

fisa\_id NUMBER(5),

cititor\_id NUMBER(5) CONSTRAINT null\_cititor\_fisa not null,

carte\_id NUMBER(5) CONSTRAINT null\_carte\_fisa not null,

biblioteca\_id NUMBER(5) CONSTRAINT null\_biblioteca\_fisa not null,

permis\_id NUMBER(5) CONSTRAINT null\_permis\_fisa not null,

data\_imprumut DATE DEFAULT sysdate,

data\_restituire DATE DEFAULT sysdate,

CONSTRAINT pk\_fisa PRIMARY KEY(fisa\_id),

CONSTRAINT fk\_cititor\_fisa FOREIGN KEY(cititor\_id) REFERENCES CITITORI(cititor\_id) ON DELETE CASCADE,

CONSTRAINT fk\_carte\_fisa FOREIGN KEY(carte\_id) REFERENCES CARTI(carte\_id) ON DELETE CASCADE,

CONSTRAINT fk\_biblioteca\_fisa FOREIGN KEY(biblioteca\_id) REFERENCES BIBLIOTECI(biblioteca\_id) ON DELETE CASCADE,

CONSTRAINT fk\_permis\_fisa FOREIGN KEY(permis\_id) REFERENCES PERMIS\_DE\_INTRARE(permis\_id) ON DELETE CASCADE,

CONSTRAINT data\_logic\_fisa CHECK(data\_imprumut <= data\_restituire),

CONSTRAINT unq\_carte\_data\_imprumut\_fisa UNIQUE(cititor\_id, carte\_id, data\_imprumut)

);

COMMIT;

CREATE TABLE FURNIZORI(

furnizor\_id NUMBER(5),

denumire VARCHAR2(30) CONSTRAINT null\_denumire\_furn not null,

telefon VARCHAR2(20) CONSTRAINT null\_telefon\_furn not null,

email VARCHAR2(320) CONSTRAINT null\_email\_furn not null,

adresa\_id NUMBER(5) CONSTRAINT null\_adresa\_furn not null,

CONSTRAINT pk\_furn PRIMARY KEY(furnizor\_id),

CONSTRAINT fk\_adresa\_furn FOREIGN KEY(adresa\_id) REFERENCES ADRESE(adresa\_id) ON DELETE CASCADE,

CONSTRAINT unq\_denumire\_furn UNIQUE(denumire),

CONSTRAINT unq\_telefon\_furn UNIQUE(telefon),

CONSTRAINT unq\_email\_furn UNIQUE(email),

CONSTRAINT lungime\_telefon\_furn CHECK(LENGTH(telefon) >= 10)

);

COMMIT;

CREATE TABLE LUCREAZA(

editura\_id NUMBER(5),

furnizor\_id NUMBER(5),

CONSTRAINT pk\_lucreaza PRIMARY KEY(editura\_id, furnizor\_id),

CONSTRAINT fk\_editura\_lucreaza FOREIGN KEY(editura\_id) REFERENCES EDITURI(editura\_id) ON DELETE CASCADE,

CONSTRAINT fk\_furnizor\_lucreaza FOREIGN KEY(furnizor\_id) REFERENCES FURNIZORI(furnizor\_id) ON DELETE CASCADE

);

COMMIT;

CREATE TABLE COMENZI(

comanda\_id NUMBER(5),

furnizor\_id NUMBER(5) CONSTRAINT null\_furnizor\_comenzi not null,

carte\_id NUMBER(5) CONSTRAINT null\_carte\_comenzi not null,

biblioteca\_id NUMBER(5) CONSTRAINT null\_biblioteca\_comenzi not null,

nr\_exemplare NUMBER(10) CONSTRAINT null\_nr\_exemplare\_comenzi not null,

pret\_comanda NUMBER(10, 2) CONSTRAINT null\_pret\_comenzi not null,

CONSTRAINT pk\_comenzi PRIMARY KEY(comanda\_id),

CONSTRAINT fk\_furnizor\_comenzi FOREIGN KEY(furnizor\_id) REFERENCES FURNIZORI(furnizor\_id) ON DELETE CASCADE,

CONSTRAINT fk\_carte\_comenzi FOREIGN KEY(carte\_id) REFERENCES CARTI(carte\_id) ON DELETE CASCADE,

CONSTRAINT fk\_biblioteca\_comenzi FOREIGN KEY(biblioteca\_id) REFERENCES BIBLIOTECI(biblioteca\_id) ON DELETE CASCADE,

CONSTRAINT nr\_exemplare\_logic CHECK(nr\_exemplare > 0),

CONSTRAINT pret\_logic CHECK(pret\_comanda >= 0)

);

COMMIT;

CREATE TABLE SUBTIPURI(

subtip\_id NUMBER(5),

denumire VARCHAR2(30) CONSTRAINT null\_denumire\_subtipuri not null,

CONSTRAINT pk\_subtipuri PRIMARY KEY(subtip\_id),

CONSTRAINT unq\_denumire\_subtipuri UNIQUE(denumire)

);

COMMIT;

CREATE TABLE CATALOGARE(

carte\_id NUMBER(5),

subtip\_id NUMBER(5),

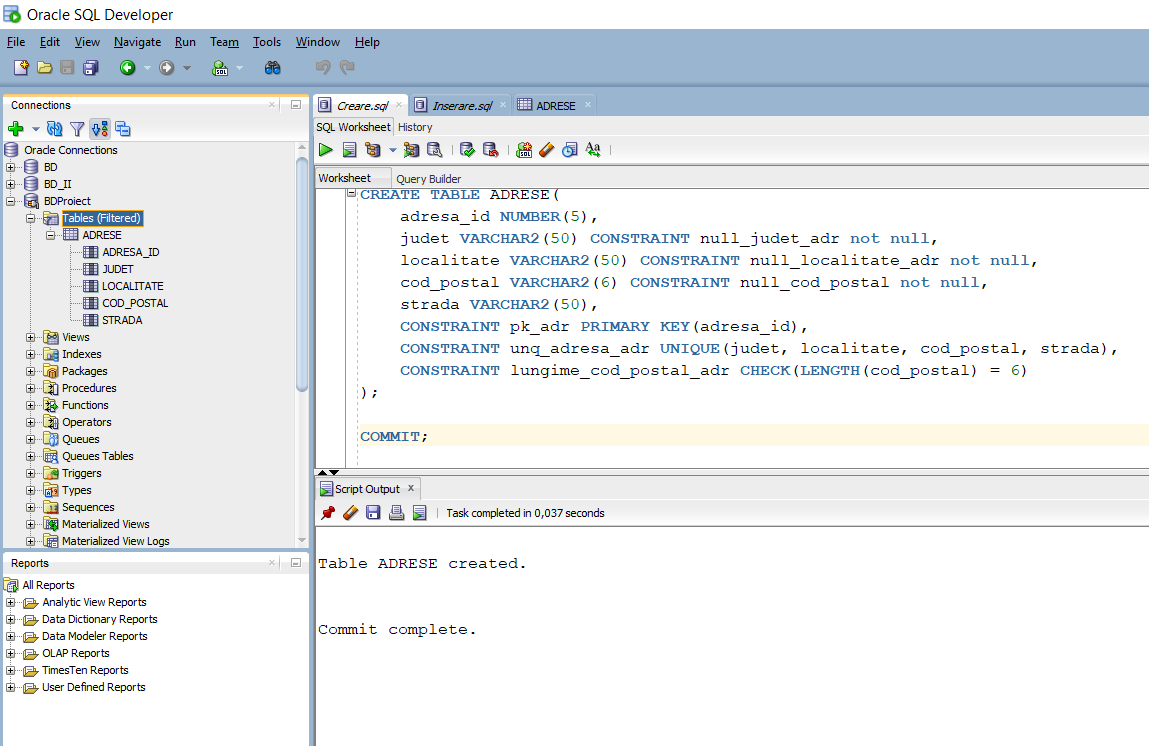
CONSTRAINT pk\_catalog PRIMARY KEY(carte\_id, subtip\_id),

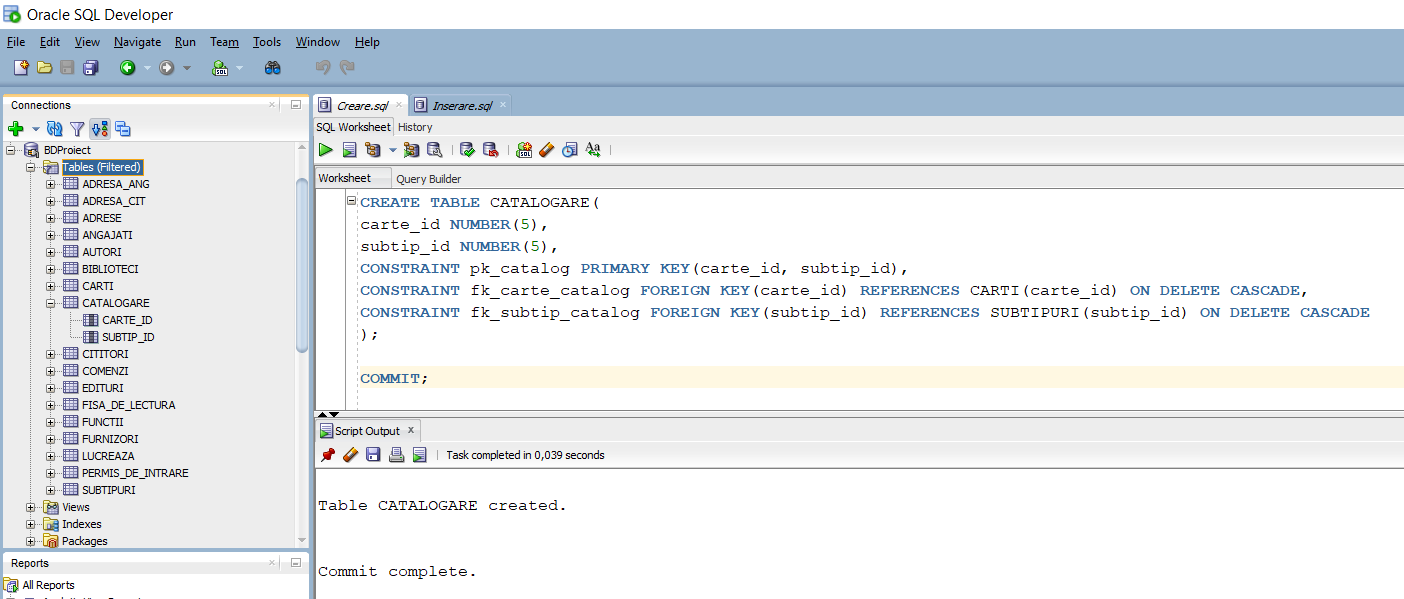
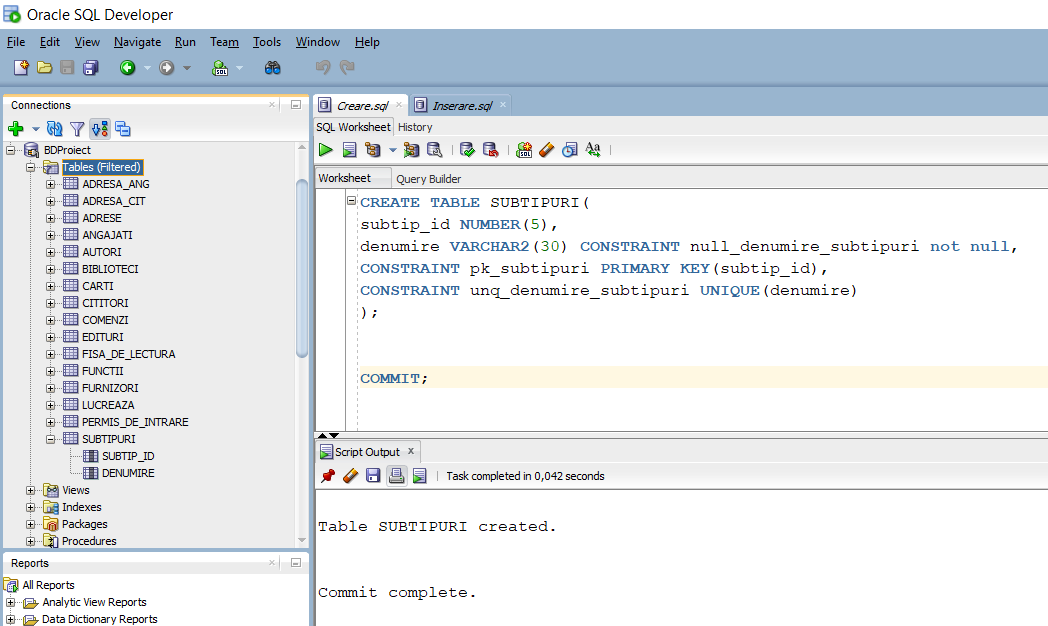
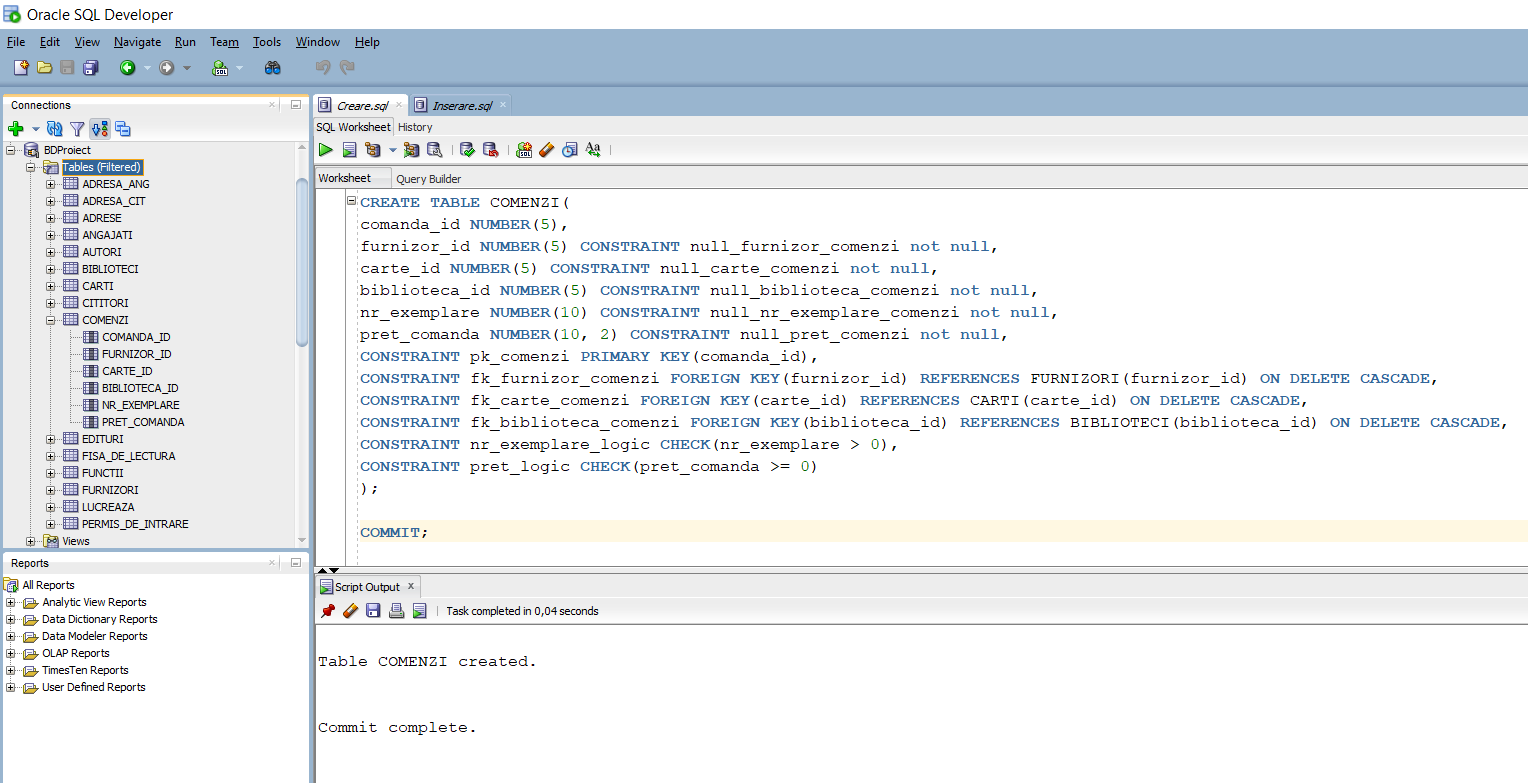
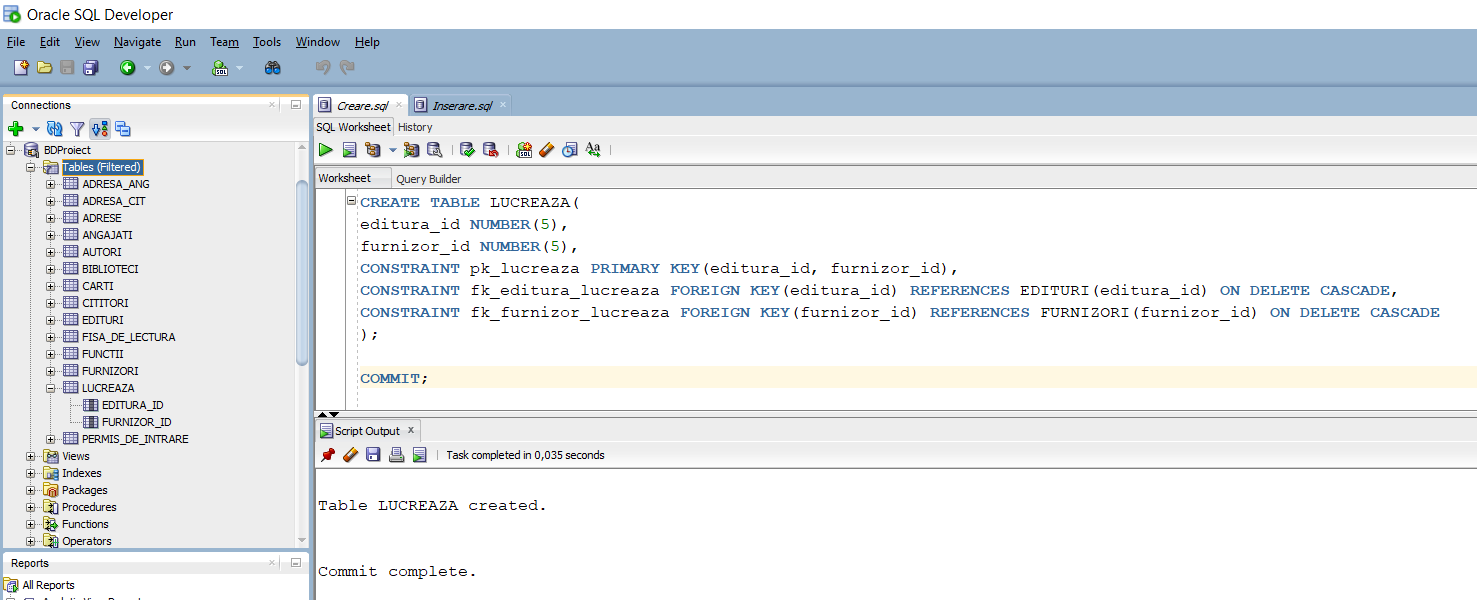
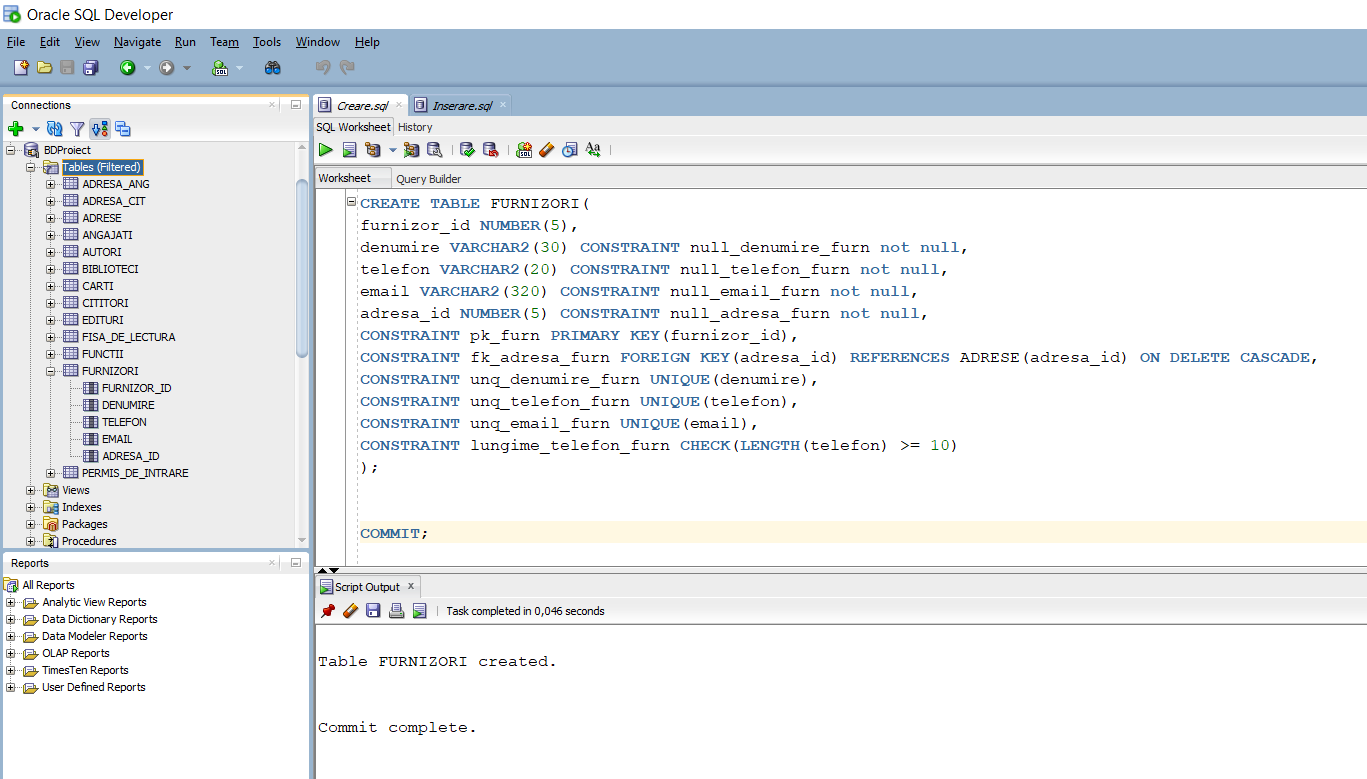
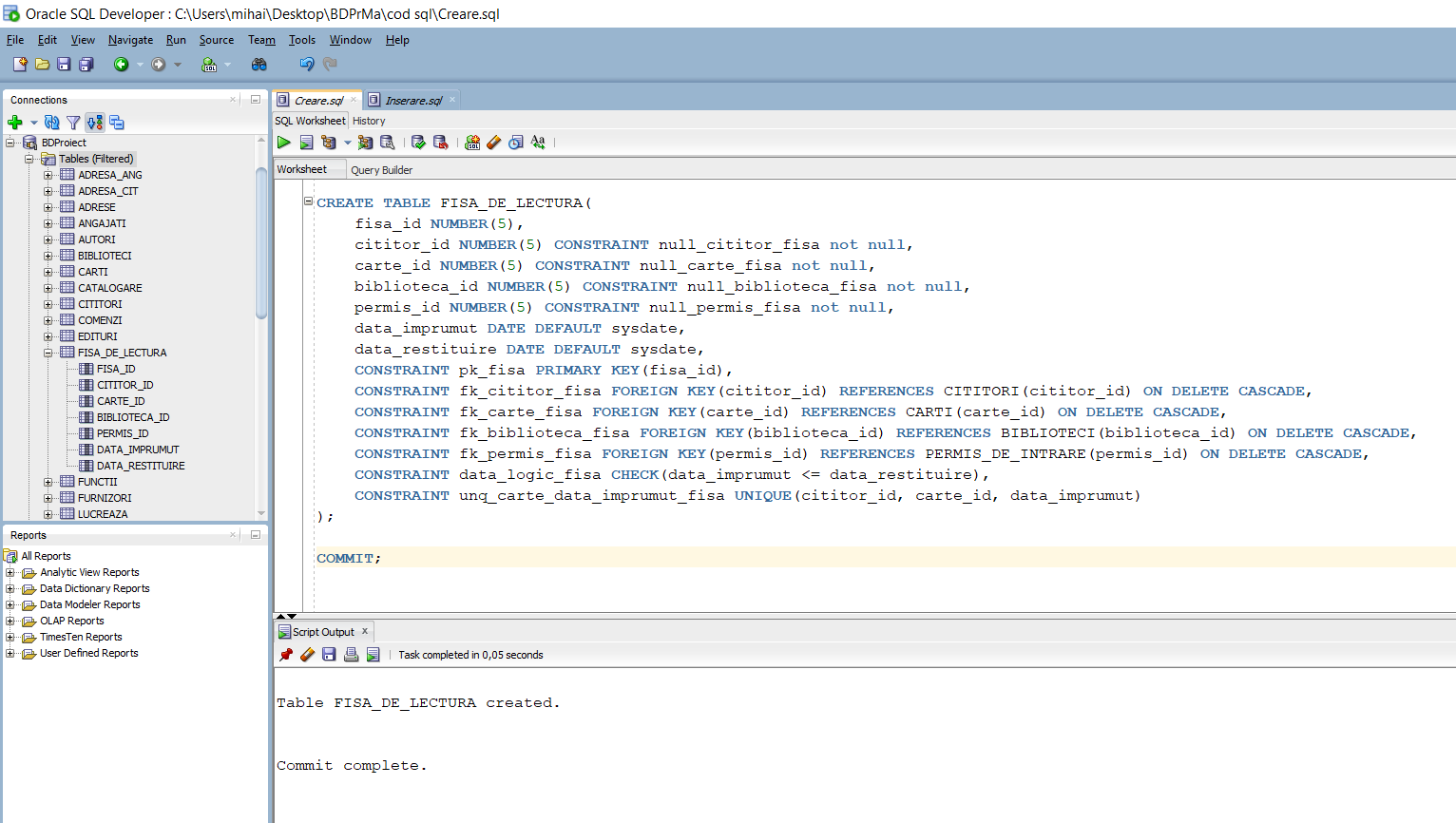
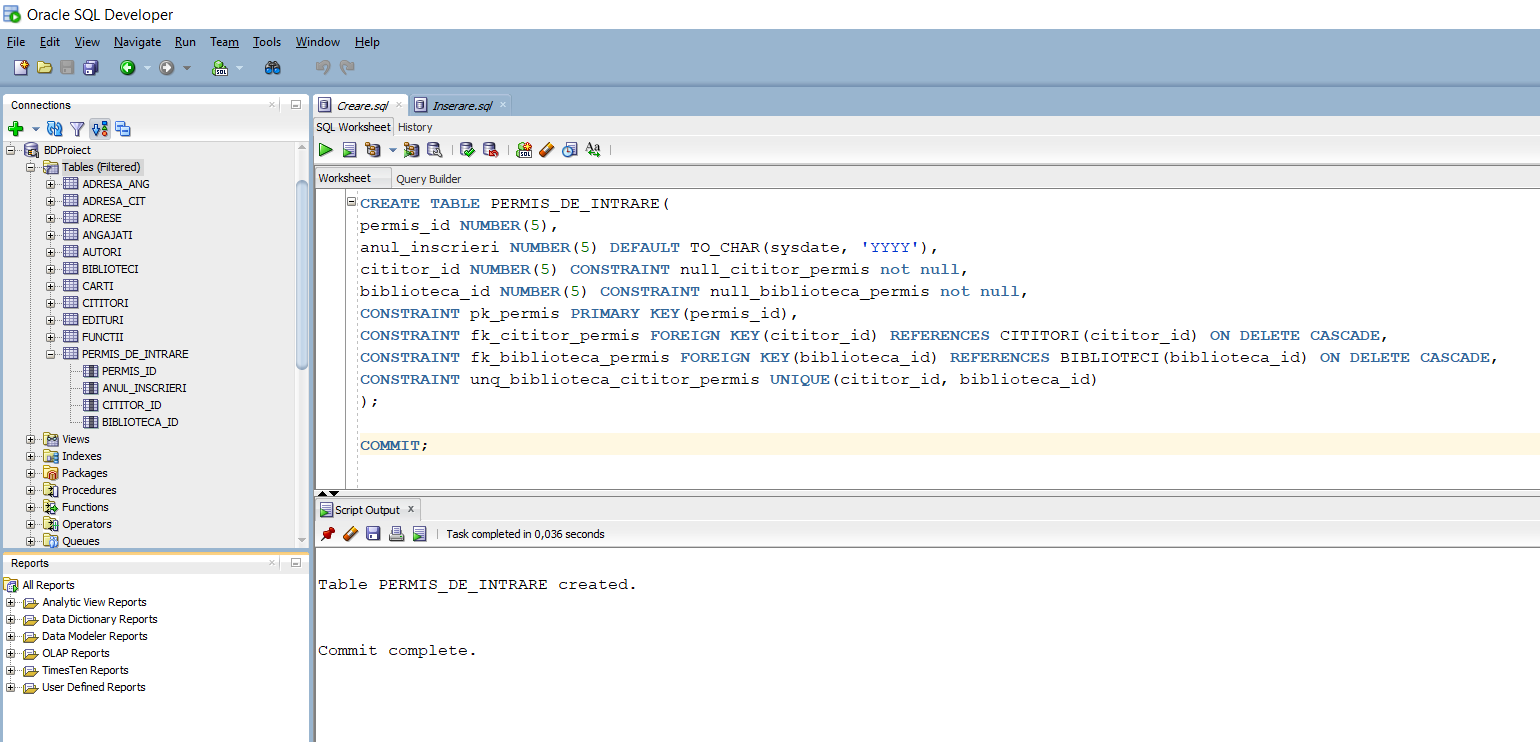
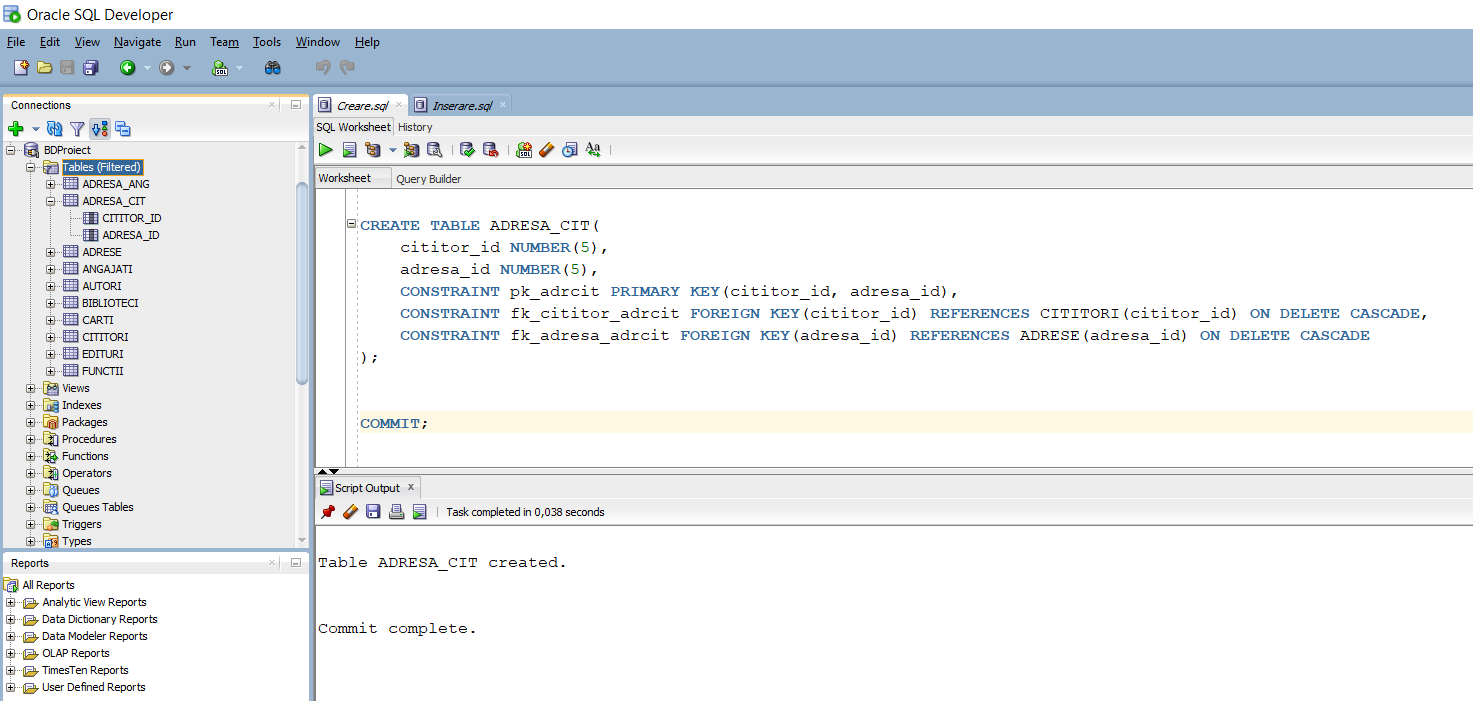
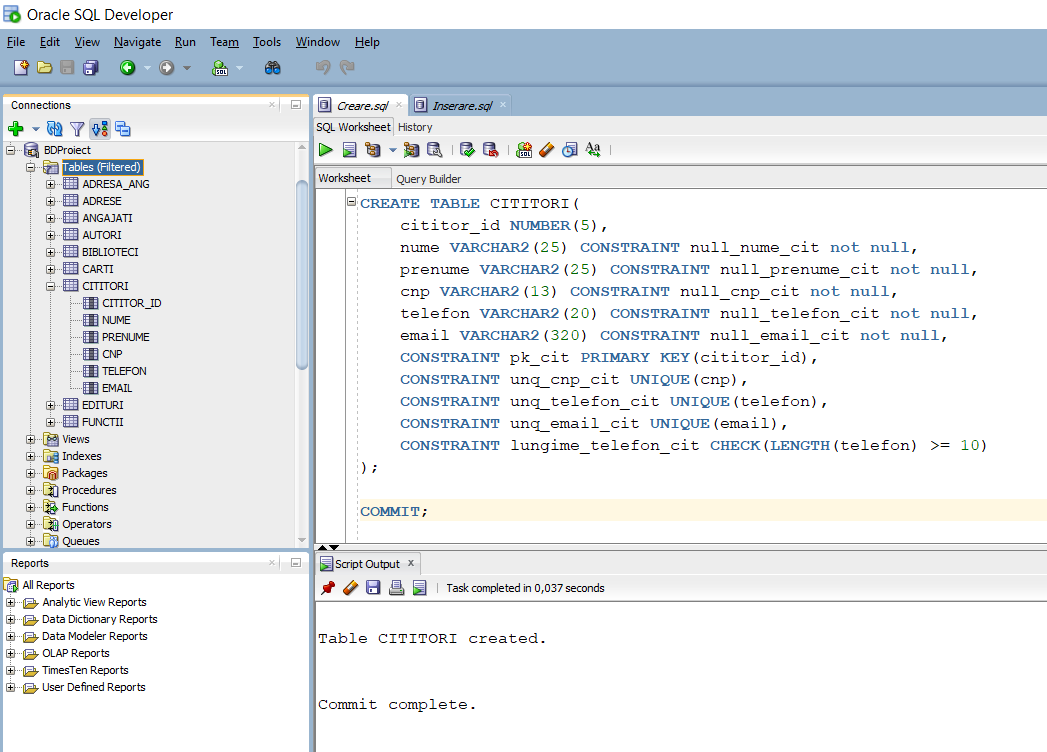
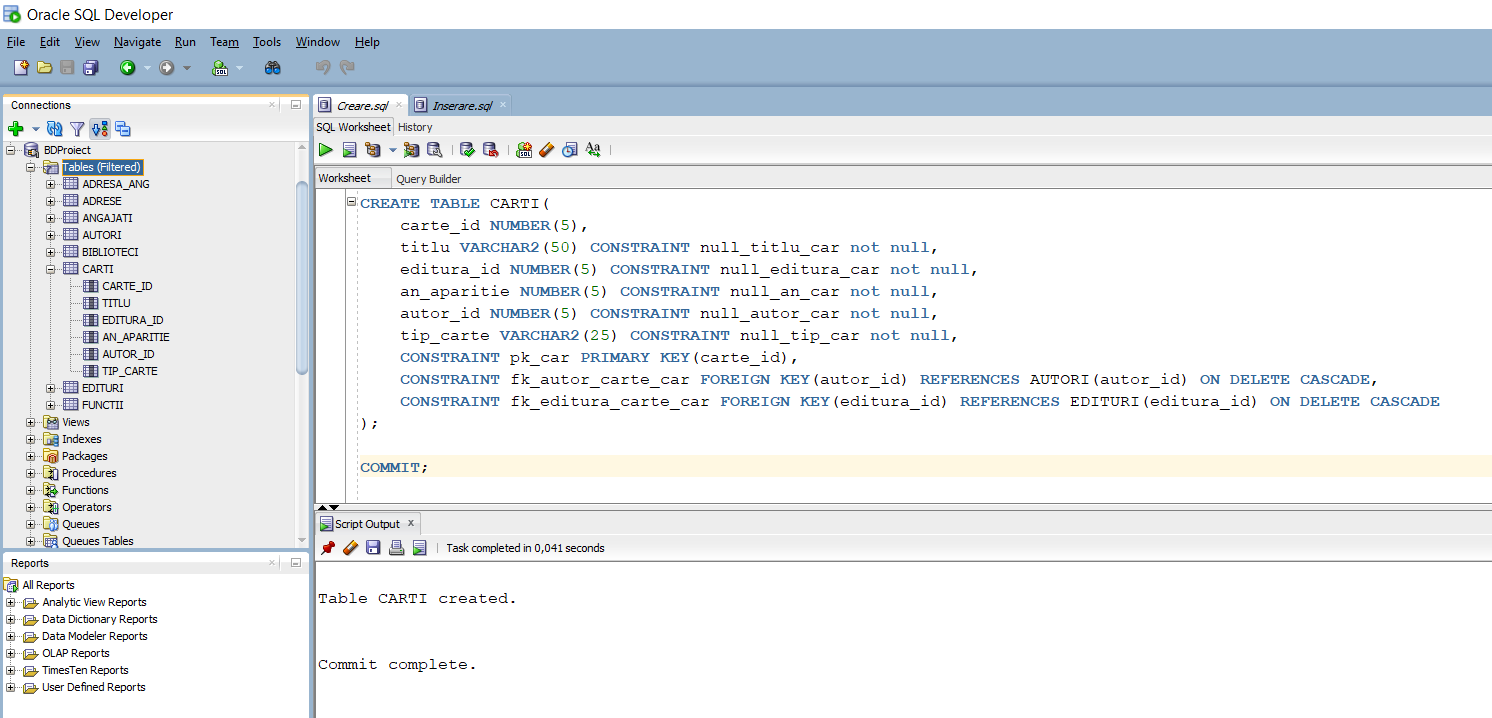
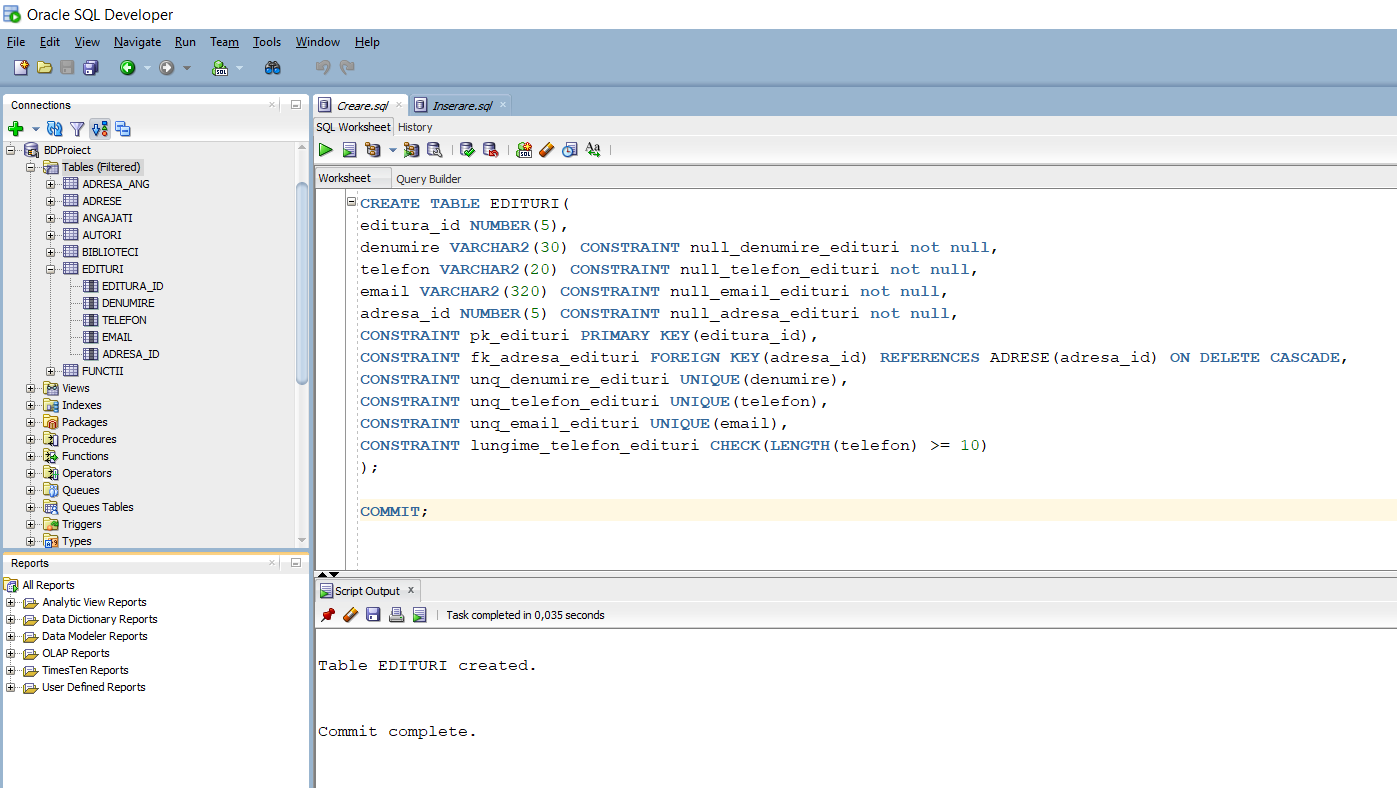
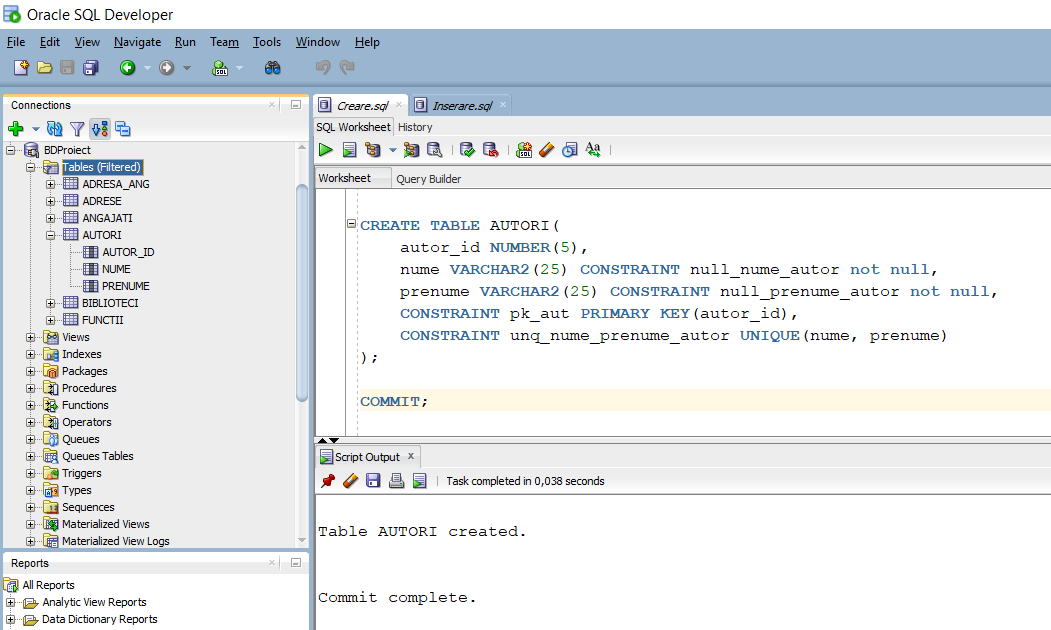
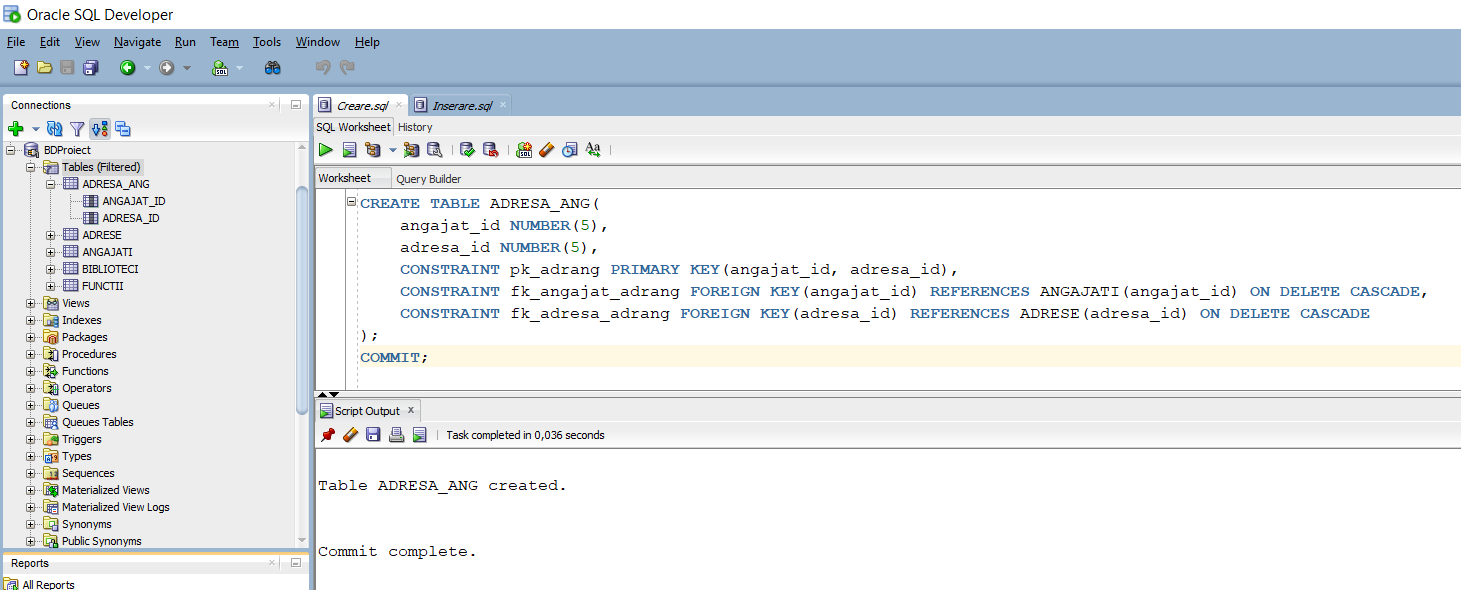
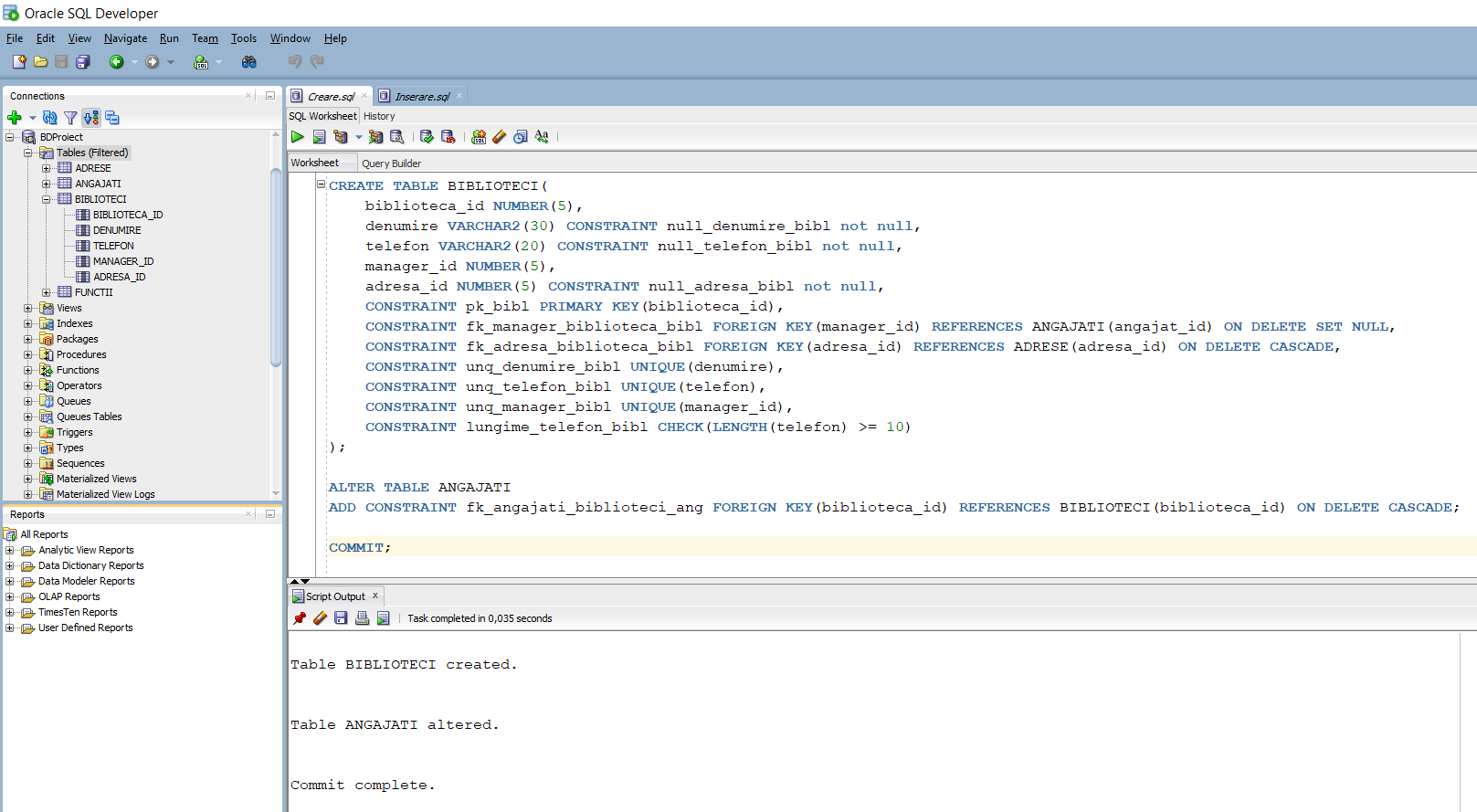
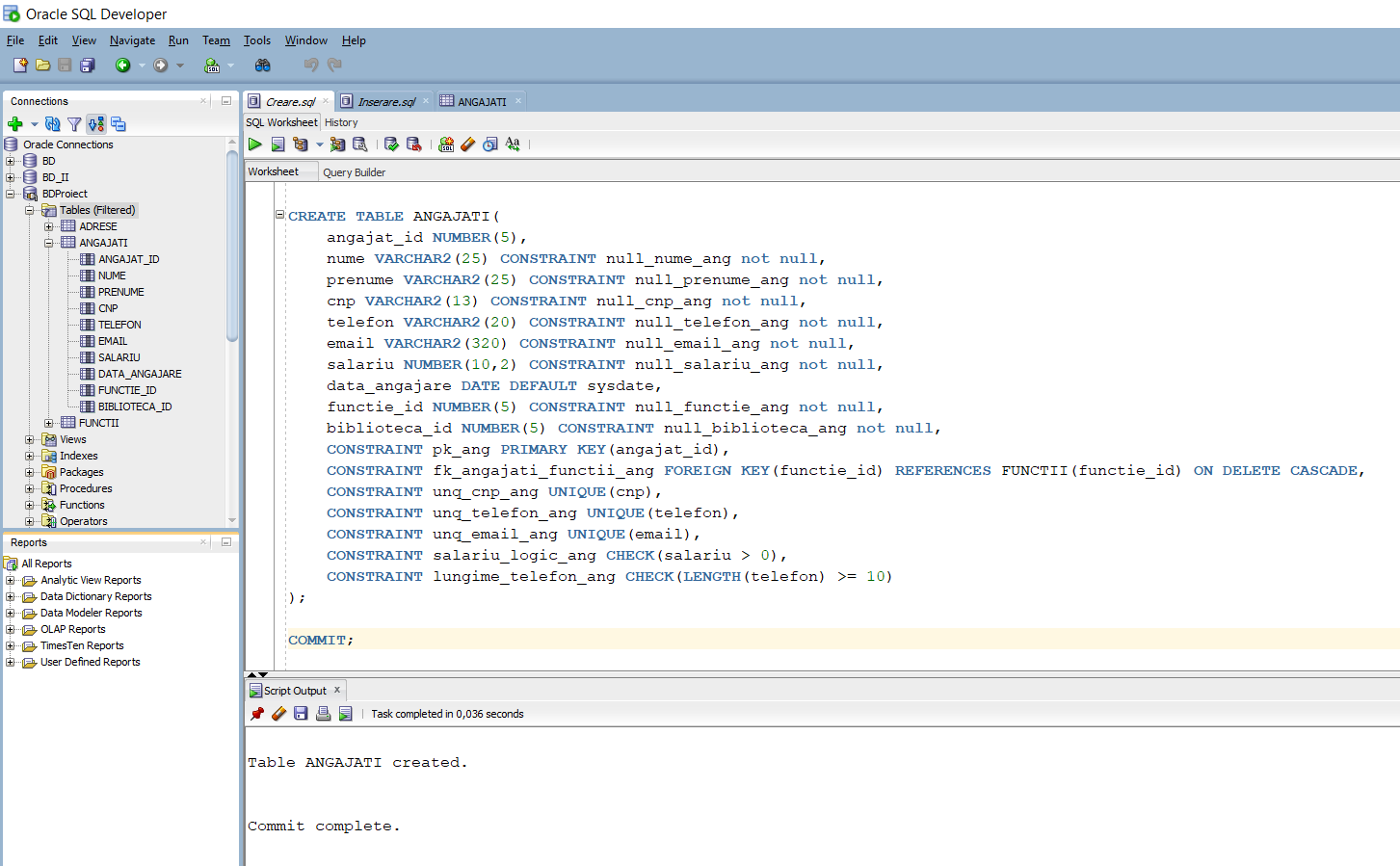
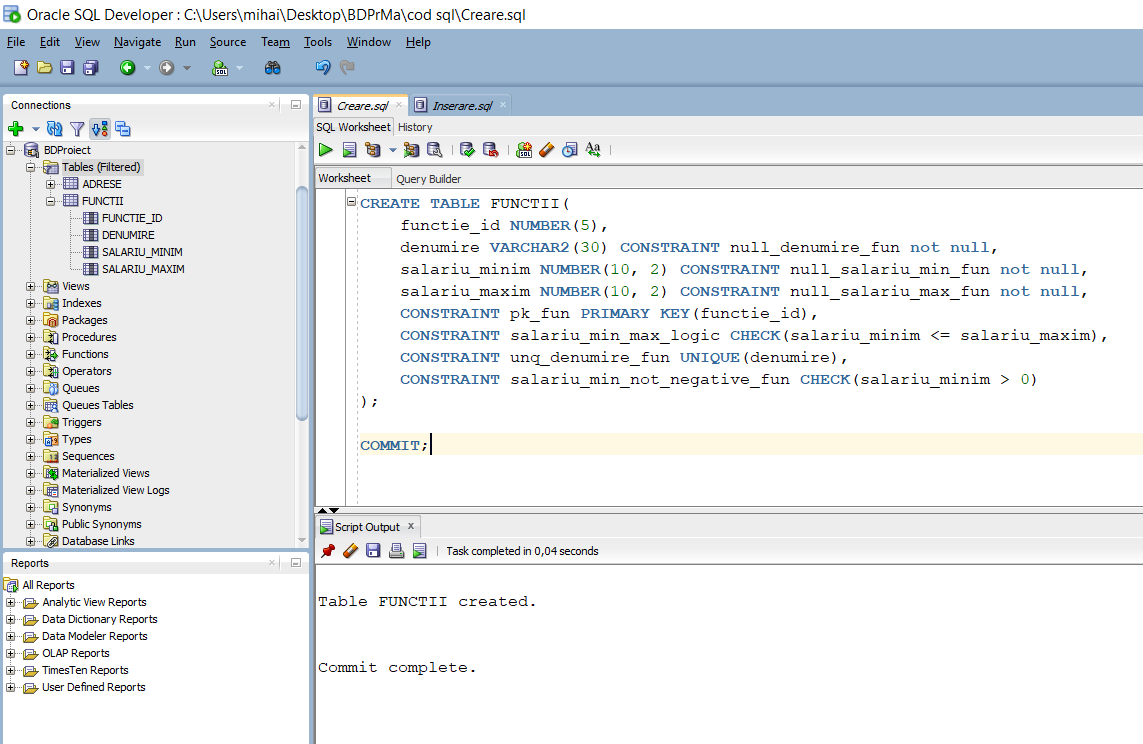
CONSTRAINT fk\_carte\_catalog FOREIGN KEY(carte\_id) REFERENCES CARTI(carte\_id) ON DELETE CASCADE,

CONSTRAINT fk\_subtip\_catalog FOREIGN KEY(subtip\_id) REFERENCES SUBTIPURI(subtip\_id) ON DELETE CASCADE

);

COMMIT;





**5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).**

CREATE SEQUENCE seq\_adrese

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_angajati

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_autori

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_biblioteci

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_carti

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_cititori

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_comenzi

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_edituri

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_fisa\_de\_lectura

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_functii

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_furnizori

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_permis\_de\_intrare

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

CREATE SEQUENCE seq\_subtipuri

START WITH 1

INCREMENT BY 1

NOCYCLE

NOCACHE;

COMMIT;

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Alba', 'Sebe?', '515800', 'Bistrei');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Olt', 'Brastavă?u', '237045', 'Bujorului');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Cluj', 'Cluj-Napoca', '400417', 'Unirii');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Bihor', 'Oradea', '410087', 'Bacăului');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Sibiu', 'Turnu Ro?u', '557285', 'Olte?');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Dolj', 'Popoveni', '200003', 'Brăila');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Maramure?', 'Baia Mare', '430301', 'Vasile Alecsandri');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Gorj', 'Târgu Jiu', '210101', 'Aleea Livezi');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Satu Mare', 'Satu Mare', '440141', 'Dariu Pop');

INSERT INTO ADRESE(adresa\_id, judet, localitate, cod\_postal, strada)

VALUES (SEQ\_ADRESE.nextval, 'Ia?i', 'Ia?i', '700341', 'Vasile Lupu');

COMMIT;

INSERT INTO FUNCTII(functie\_id, denumire, salariu\_minim, salariu\_maxim)

VALUES (SEQ\_FUNCTII.nextval, 'Bibliotecar', 6000, 12000);

INSERT INTO FUNCTII(functie\_id, denumire, salariu\_minim, salariu\_maxim)

VALUES (SEQ\_FUNCTII.nextval, 'Pază', 1000, 5000);

INSERT INTO FUNCTII(functie\_id, denumire, salariu\_minim, salariu\_maxim)

VALUES (SEQ\_FUNCTII.nextval, 'Catalogare', 3000, 9000);

INSERT INTO FUNCTII(functie\_id, denumire, salariu\_minim, salariu\_maxim)

VALUES (SEQ\_FUNCTII.nextval, 'Consilier', 4000, 10000);

INSERT INTO FUNCTII(functie\_id, denumire, salariu\_minim, salariu\_maxim)

VALUES (SEQ\_FUNCTII.nextval, 'Marketing', 6000, 8000);

COMMIT;

INSERT INTO BIBLIOTECI(biblioteca\_id, denumire, telefon, manager\_id, adresa\_id)

VALUES (SEQ\_BIBLIOTECI.nextval, 'Biblioteca Ro?ie', '0773.245.664', null, 5);

INSERT INTO BIBLIOTECI(biblioteca\_id, denumire, telefon, manager\_id, adresa\_id)

VALUES (SEQ\_BIBLIOTECI.nextval, 'Biblioteca De Aur', '0773.236.918', null, 9);

INSERT INTO BIBLIOTECI(biblioteca\_id, denumire, telefon, manager\_id, adresa\_id)

VALUES (SEQ\_BIBLIOTECI.nextval, 'Biblioteca Mihai Eminescu', '0772.019.467', null, 2);

INSERT INTO BIBLIOTECI(biblioteca\_id, denumire, telefon, manager\_id, adresa\_id)

VALUES (SEQ\_BIBLIOTECI.nextval, 'Biblioteca Bacovia', '0772.162.250', null, 4);

INSERT INTO BIBLIOTECI(biblioteca\_id, denumire, telefon, manager\_id, adresa\_id)

VALUES (SEQ\_BIBLIOTECI.nextval, 'Biblioteca Ion Creangă', '0772.309.337', null, 5);

COMMIT;

INSERT INTO ANGAJATI(angajat\_id, nume, prenume, cnp, telefon, email,

salariu, data\_angajare, functie\_id, biblioteca\_id)

VALUES (SEQ\_ANGAJATI.nextval, 'Petran', 'Teodor', '2476281827462',

'0772.124.547', 'PetranTeodor@gmail.com', 10000, '27/01/1944', 1, 4);

INSERT INTO ANGAJATI(angajat\_id, nume, prenume, cnp, telefon, email,

salariu, data\_angajare, functie\_id, biblioteca\_id)

VALUES (SEQ\_ANGAJATI.nextval, 'Funar', 'Ioan', '1277364727462',

'0772.754.124', 'IoanFunar@gmail.com', 7000, '20/09/1985', 4, 3);

INSERT INTO ANGAJATI(angajat\_id, nume, prenume, cnp, telefon, email,

salariu, data\_angajare, functie\_id, biblioteca\_id)

VALUES (SEQ\_ANGAJATI.nextval, 'David', 'Alexandru', '1007264727890',

'0772.493.992', 'DavidAlexandru@gmail.com', 6500, '08/06/1999', 5, 5);

INSERT INTO ANGAJATI(angajat\_id, nume, prenume, cnp, telefon, email,

salariu, data\_angajare, functie\_id, biblioteca\_id)

VALUES (SEQ\_ANGAJATI.nextval, 'Ungur', 'Lucia', '1009805727890',

'0772.586.021', 'UngurLucia@gmail.com', 6000, '07/12/1987', 3, 1);

INSERT INTO ANGAJATI(angajat\_id, nume, prenume, cnp, telefon, email,

salariu, data\_angajare, functie\_id, biblioteca\_id)

VALUES (SEQ\_ANGAJATI.nextval, 'Cojocaru', 'Virgil', '9992264773829',

'0772.235.754', 'CojocaruVirgil@gmail.com', 2500, '10/05/1980', 2, 2);

COMMIT;

UPDATE biblioteci

SET manager\_id = biblioteca\_id

where MOD(biblioteca\_id, 2) = 0;

COMMIT;

INSERT INTO CITITORI(cititor\_id, nume, prenume, cnp, telefon, email)

VALUES (SEQ\_CITITORI.nextval, 'Lupu', 'Ligia', '1234567890123', '0772.938.856',

'LigiaLupu@gmail.com');

INSERT INTO CITITORI(cititor\_id, nume, prenume, cnp, telefon, email)

VALUES (SEQ\_CITITORI.nextval, 'Văduva', 'Adela', '1234554390123', '0772.845.584',

'VaduvaAdela@gmail.com');

INSERT INTO CITITORI(cititor\_id, nume, prenume, cnp, telefon, email)

VALUES (SEQ\_CITITORI.nextval, 'Vasile', 'Dorian', '1784567890123', '0772.125.797',

'DorianVasile@gmail.com');

INSERT INTO CITITORI(cititor\_id, nume, prenume, cnp, telefon, email)

VALUES (SEQ\_CITITORI.nextval, 'Solomon', 'Ivan', '1237777790123', '0772.999.856',

'SolomonIvan@gmail.com');

INSERT INTO CITITORI(cititor\_id, nume, prenume, cnp, telefon, email)

VALUES (SEQ\_CITITORI.nextval, 'Vasile', 'Andrada', '1876567658423', '0772.788.100',

'AndradaVasile@gmail.com');

COMMIT;

INSERT INTO AUTORI(autor\_id, nume, prenume)

VALUES(SEQ\_AUTORI.nextval, 'Morris', 'Nikolina');

INSERT INTO AUTORI(autor\_id, nume, prenume)

VALUES(SEQ\_AUTORI.nextval, 'Tatham', 'Aishwarya');

INSERT INTO AUTORI(autor\_id, nume, prenume)

VALUES(SEQ\_AUTORI.nextval, 'Brambilla', 'Shprintze');

INSERT INTO AUTORI(autor\_id, nume, prenume)

VALUES(SEQ\_AUTORI.nextval, 'Zając', 'Hayder');

INSERT INTO AUTORI(autor\_id, nume, prenume)

VALUES(SEQ\_AUTORI.nextval, 'Modugno', 'Loukianos');

COMMIT;

INSERT INTO EDITURI(editura\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_EDITURI.nextval, 'Editura Roll', '0773.244.533', 'EdituraRoll@gmail.com', 5);

INSERT INTO EDITURI(editura\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_EDITURI.nextval, 'Editura Sketch', '0770.214.654', 'EdituraSketch@gmail.com', 1);

INSERT INTO EDITURI(editura\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_EDITURI.nextval, 'Editura Lot', '0770.252.647', 'EdituraLot@gmail.com', 2);

INSERT INTO EDITURI(editura\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_EDITURI.nextval, 'Editura Thesis', '0770.241.447', 'EdituraThesis@gmail.com', 6);

INSERT INTO EDITURI(editura\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_EDITURI.nextval, 'Editura Arena', '0774.123.347', 'EdituraArena@gmail.com', 10);

COMMIT;

INSERT INTO FURNIZORI(furnizor\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_FURNIZORI.nextval, 'Turkey Cucumber', '0774.004.122', 'TurkeyCucumber@gmail.com', 7);

INSERT INTO FURNIZORI(furnizor\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_FURNIZORI.nextval, 'Tradition Spill', '0774.353.689', 'TraditionSpill@gmail.com', 1);

INSERT INTO FURNIZORI(furnizor\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_FURNIZORI.nextval, 'Mobile Feather', '0774.121.224', 'MobileFeather@gmail.com', 2);

INSERT INTO FURNIZORI(furnizor\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_FURNIZORI.nextval, 'Gallery Lion', '0774.233.646', 'GalleryLion@gmail.com', 8);

INSERT INTO FURNIZORI(furnizor\_id, denumire, telefon, email, adresa\_id)

VALUES (SEQ\_FURNIZORI.nextval, 'Gold Page', '0774.241.366', 'GoldPage@gmail.com', 5);

COMMIT;

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Animale');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Matematcă');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Fizică');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Muzică');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Artă');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Informatică');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Horror');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Drama');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Ac?iune');

INSERT INTO SUBTIPURI(subtip\_id, denumire)

VALUES (SEQ\_SUBTIPURI.nextval, 'Poli?iste');

COMMIT;

INSERT INTO CARTI(carte\_id, titlu, editura\_id, an\_aparitie, autor\_id, tip\_carte)

VALUES (SEQ\_CARTI.nextval, 'Cunoscutul', 5, 1800, 2, '?tiin?ific');

INSERT INTO CARTI(carte\_id, titlu, editura\_id, an\_aparitie, autor\_id, tip\_carte)

VALUES (SEQ\_CARTI.nextval, 'Animale terestre', 1, 1950, 5, 'Pentru copii');

INSERT INTO CARTI(carte\_id, titlu, editura\_id, an\_aparitie, autor\_id, tip\_carte)

VALUES (SEQ\_CARTI.nextval, 'Aventurile lui Ro?u', 4, 1877, 3, 'Pove?ti');

INSERT INTO CARTI(carte\_id, titlu, editura\_id, an\_aparitie, autor\_id, tip\_carte)

VALUES (SEQ\_CARTI.nextval, 'Matematică pentru începători', 2, 2000, 1, '?tiin?ific');

INSERT INTO CARTI(carte\_id, titlu, editura\_id, an\_aparitie, autor\_id, tip\_carte)

VALUES (SEQ\_CARTI.nextval, 'Poeziile naturi', 3, 1899, 4, 'Poezii');

COMMIT;

INSERT INTO COMENZI(comanda\_id, furnizor\_id, carte\_id, biblioteca\_id, nr\_exemplare, pret\_comanda)

VALUES(SEQ\_COMENZI.nextval, 4, 1, 3, 10, 500);

INSERT INTO COMENZI(comanda\_id, furnizor\_id, carte\_id, biblioteca\_id, nr\_exemplare, pret\_comanda)

VALUES(SEQ\_COMENZI.nextval, 1, 4, 5, 25, 2000);

INSERT INTO COMENZI(comanda\_id, furnizor\_id, carte\_id, biblioteca\_id, nr\_exemplare, pret\_comanda)

VALUES(SEQ\_COMENZI.nextval, 3, 2, 2, 20, 1500);

INSERT INTO COMENZI(comanda\_id, furnizor\_id, carte\_id, biblioteca\_id, nr\_exemplare, pret\_comanda)

VALUES(SEQ\_COMENZI.nextval, 5, 3, 4, 15, 1800);

INSERT INTO COMENZI(comanda\_id, furnizor\_id, carte\_id, biblioteca\_id, nr\_exemplare, pret\_comanda)

VALUES(SEQ\_COMENZI.nextval, 2, 5, 1, 30, 4000);

COMMIT;

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2000, 1, 3);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2010, 2, 5);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2001, 5, 2);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2018, 3, 1);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 1990, 4, 4);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2011, 3, 5);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2007, 4, 3);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2021, 2, 2);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2014, 1, 4);

INSERT INTO PERMIS\_DE\_INTRARE(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

VALUES (SEQ\_PERMIS\_DE\_INTRARE.nextval, 2009, 1, 5);

COMMIT;

INSERT INTO LUCREAZA(editura\_id, furnizor\_id)

VALUES(1, 2);

INSERT INTO LUCREAZA(editura\_id, furnizor\_id)

VALUES(4, 4);

INSERT INTO LUCREAZA(editura\_id, furnizor\_id)

VALUES(2, 3);

INSERT INTO LUCREAZA(editura\_id, furnizor\_id)

VALUES(5, 1);

INSERT INTO LUCREAZA(editura\_id, furnizor\_id)

VALUES(3, 5);

INSERT INTO LUCREAZA(editura\_id, furnizor\_id)

SELECT ca.editura\_id, co.furnizor\_id

FROM carti ca, comenzi co

WHERE ca.carte\_id = co.carte\_id

AND (ca.editura\_id, co.furnizor\_id) not in (

SELECT editura\_id, furnizor\_id

FROM lucreaza

);

COMMIT;

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 1, 1, 3, 1, '10/05/1980', '30/05/1980');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 2, 2, 5, 4, '01/09/1990', '11/09/1990');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 3, 5, 2, 2, '05/11/2000', '15/11/2000');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 4, 3, 1, 5, '29/09/2005', '06/10/2005');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 5, 4, 4, 3, '03/01/2010', '13/01/2010');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 6, 3, 5, 4, '09/11/2008', '19/11/2008');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 7, 4, 3, 1, '5/11/2000', '5/11/2000');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 8, 2, 2, 2, '01/09/1990', '11/09/1990');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 9, 1, 4, 3, '29/09/2005', '06/10/2005');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(SEQ\_FISA\_DE\_LECTURA.nextval, 10, 1, 5, 4, '10/05/1980', '30/05/1980');

COMMIT;

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(1, 2);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(1, 3);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(2, 1);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(2, 5);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(2, 4);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(3, 7);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(3, 8);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(3, 9);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(3, 10);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(4, 2);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(5, 1);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(5, 5);

INSERT INTO CATALOGARE(carte\_id, subtip\_id)

VALUES(5, 4);

COMMIT;

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(1, 4);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(1, 9);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(2, 10);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(2, 3);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(3, 6);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(3, 7);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(4, 9);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(4, 1);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(5, 10);

INSERT INTO ADRESA\_ANG(angajat\_id, adresa\_id)

VALUES(5, 8);

COMMIT;

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(1, 7);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(1, 10);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(2, 1);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(2, 4);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(3, 6);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(3, 3);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(4, 9);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(4, 1);

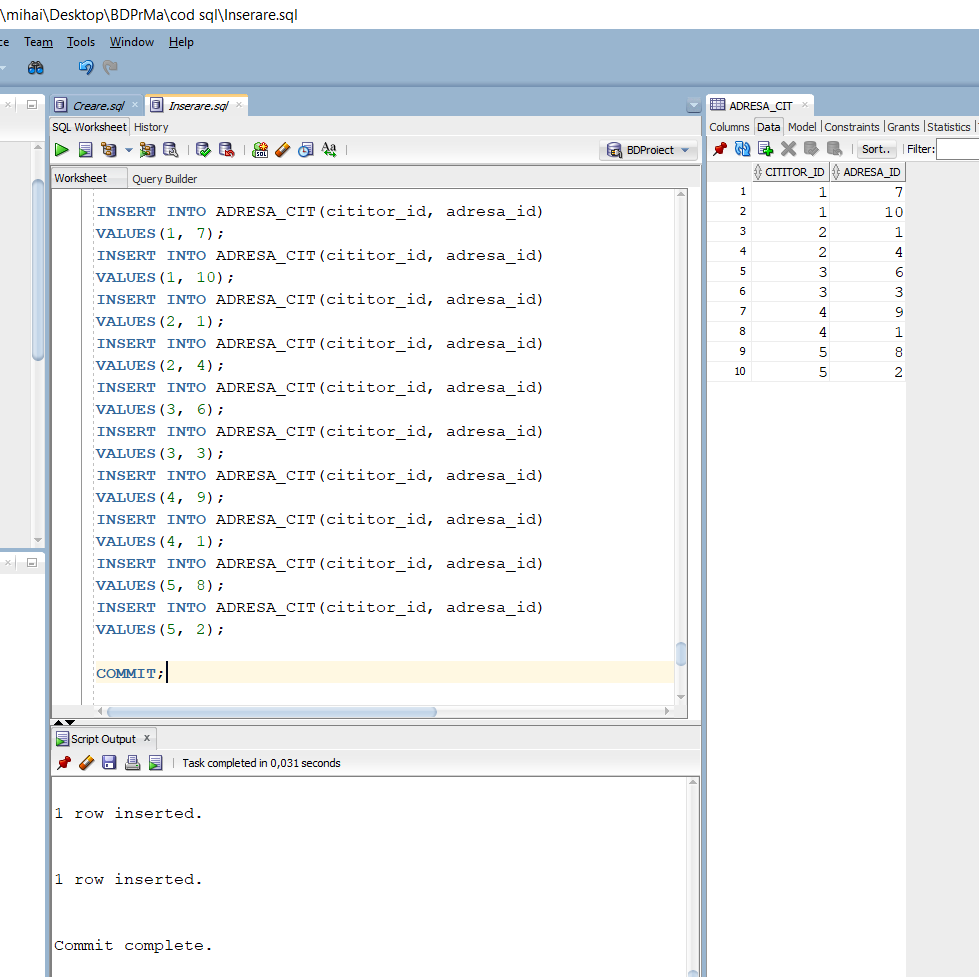
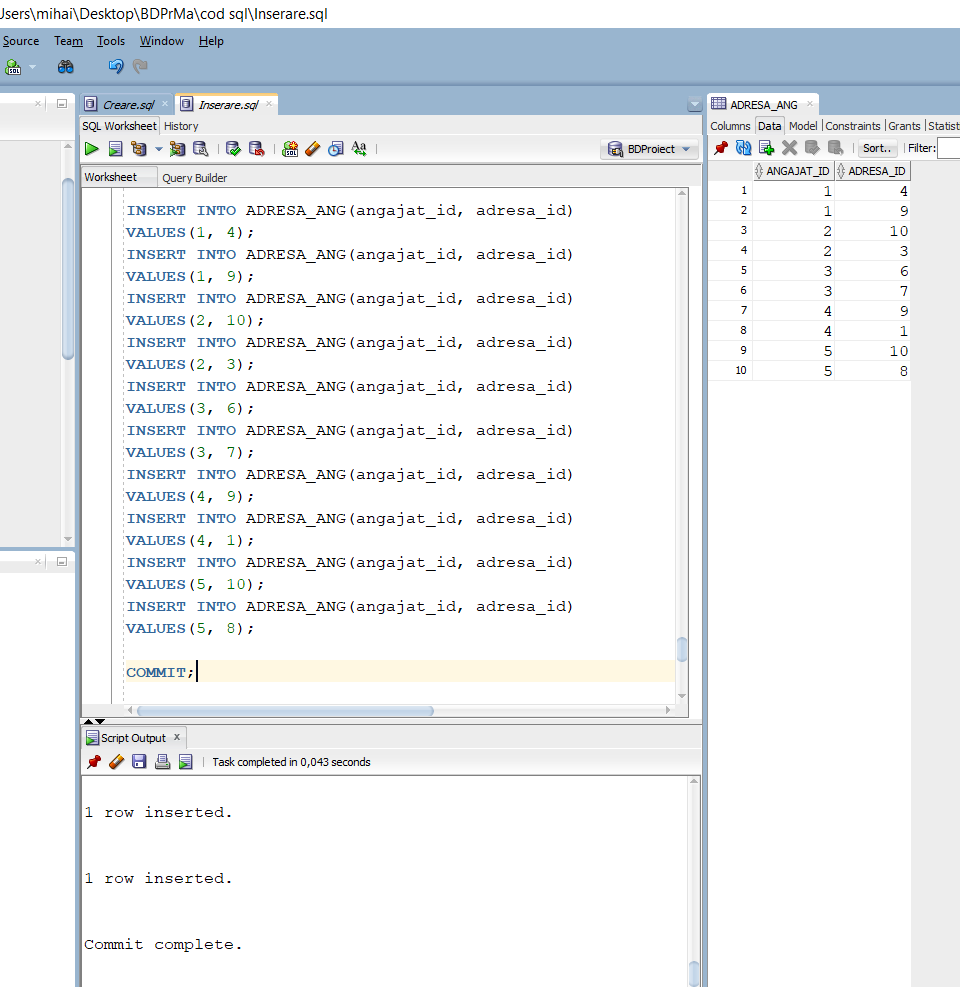
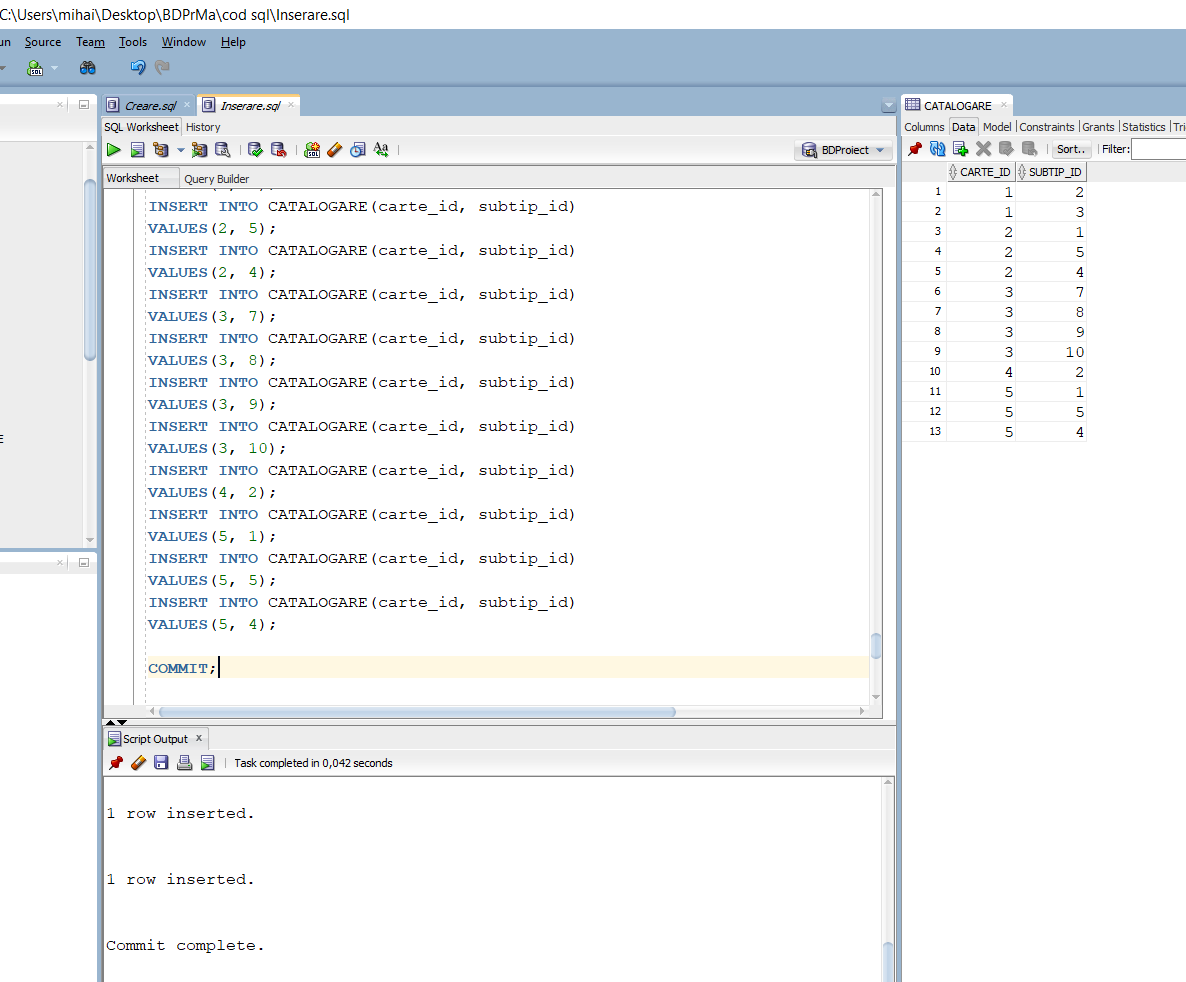
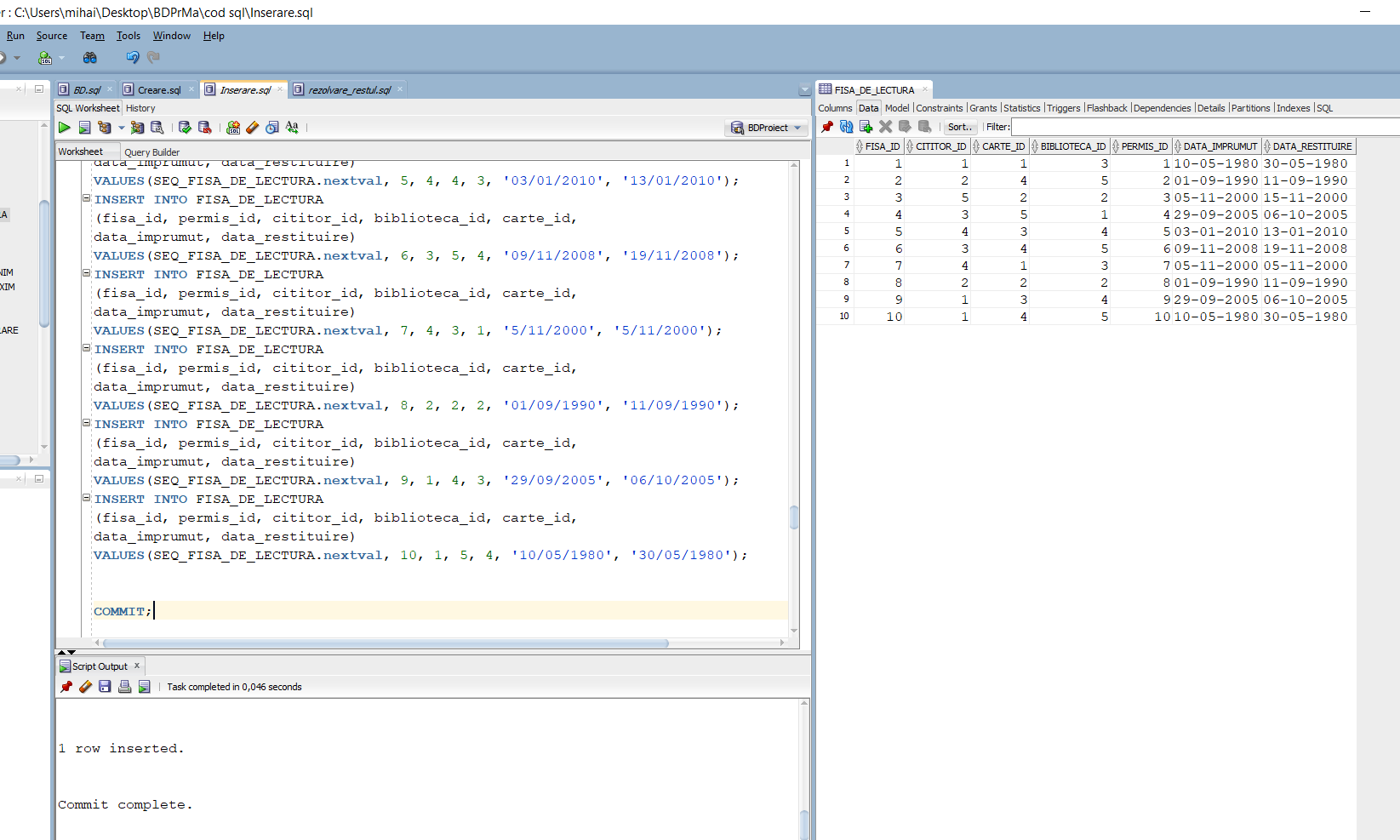
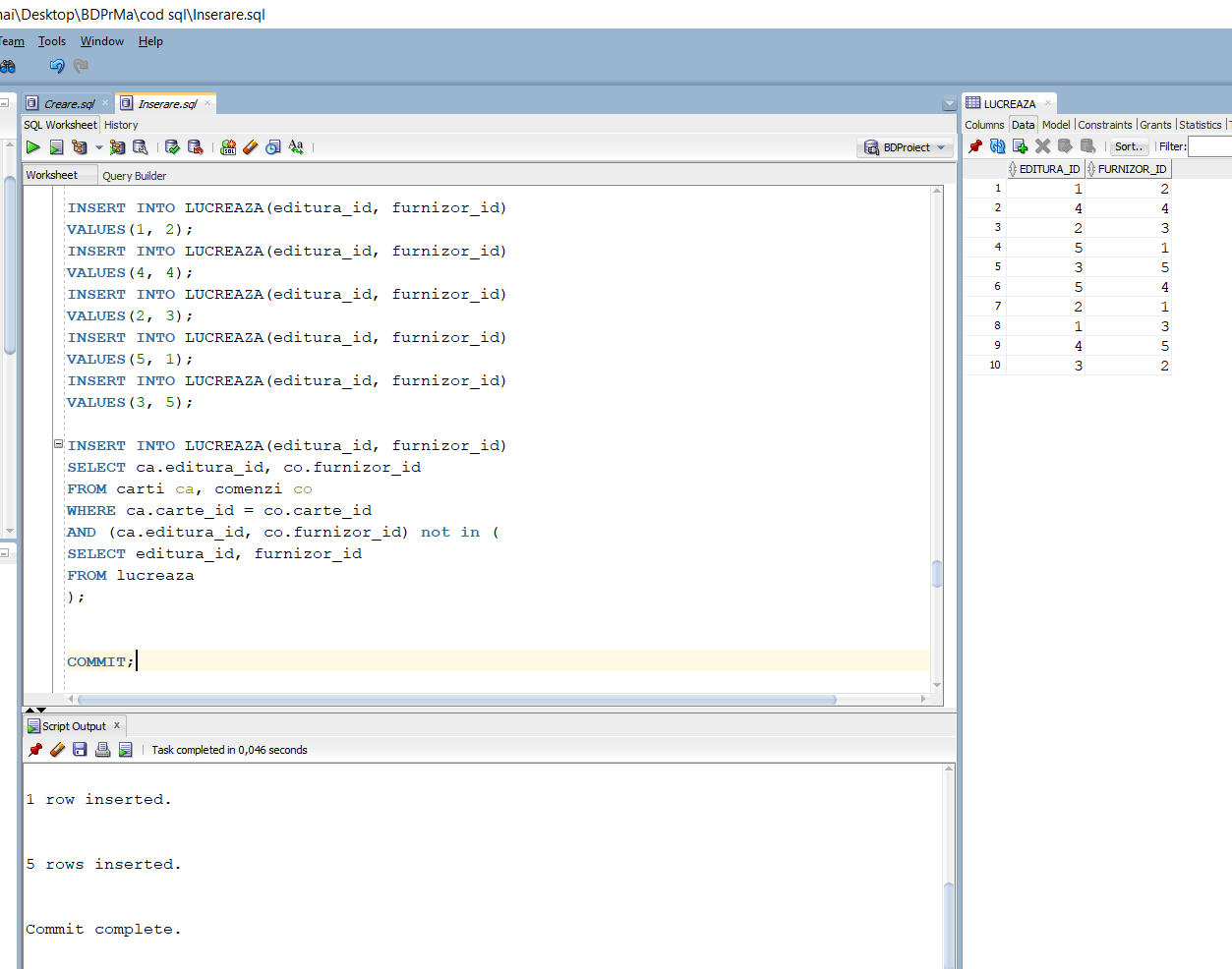
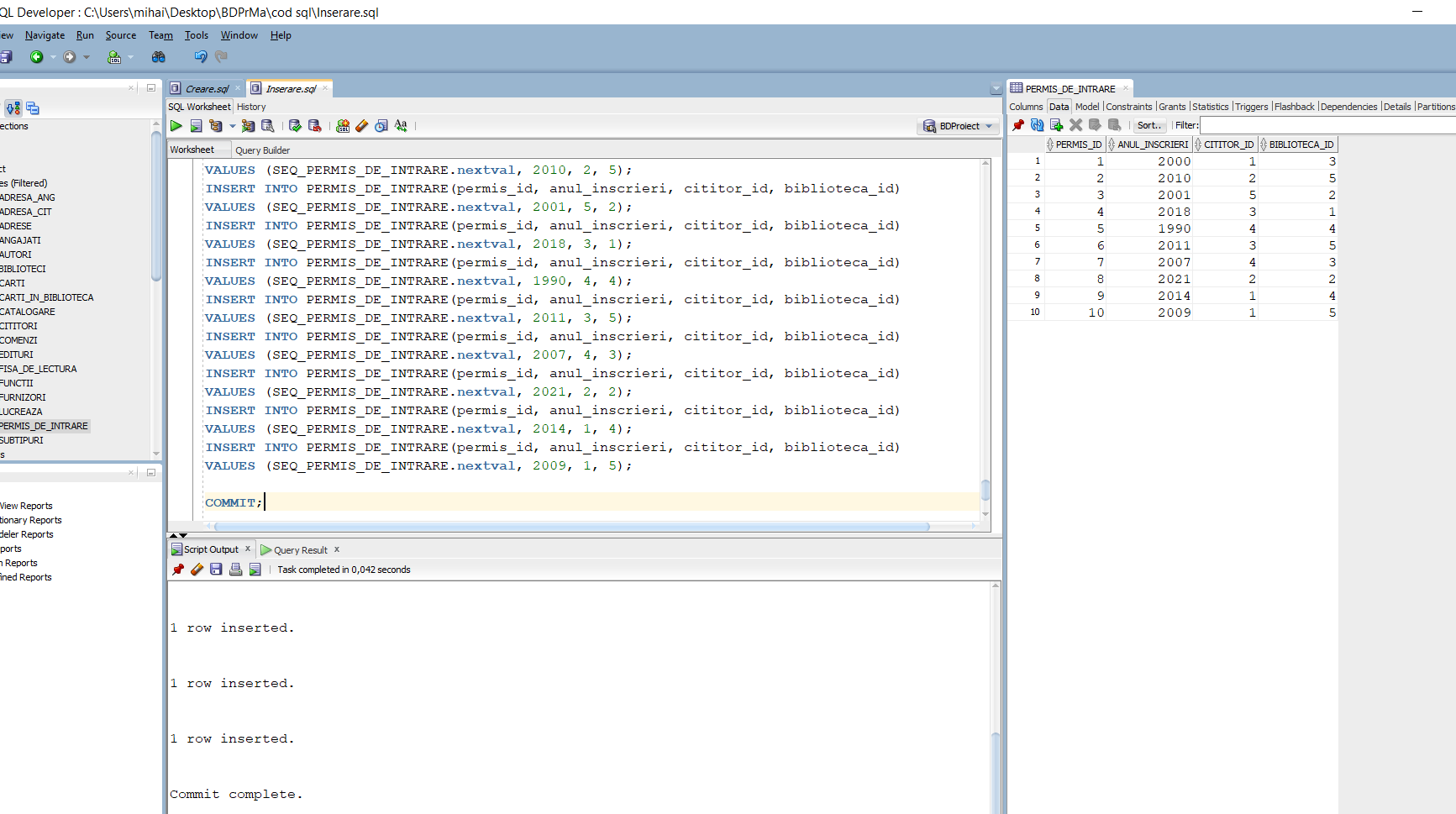
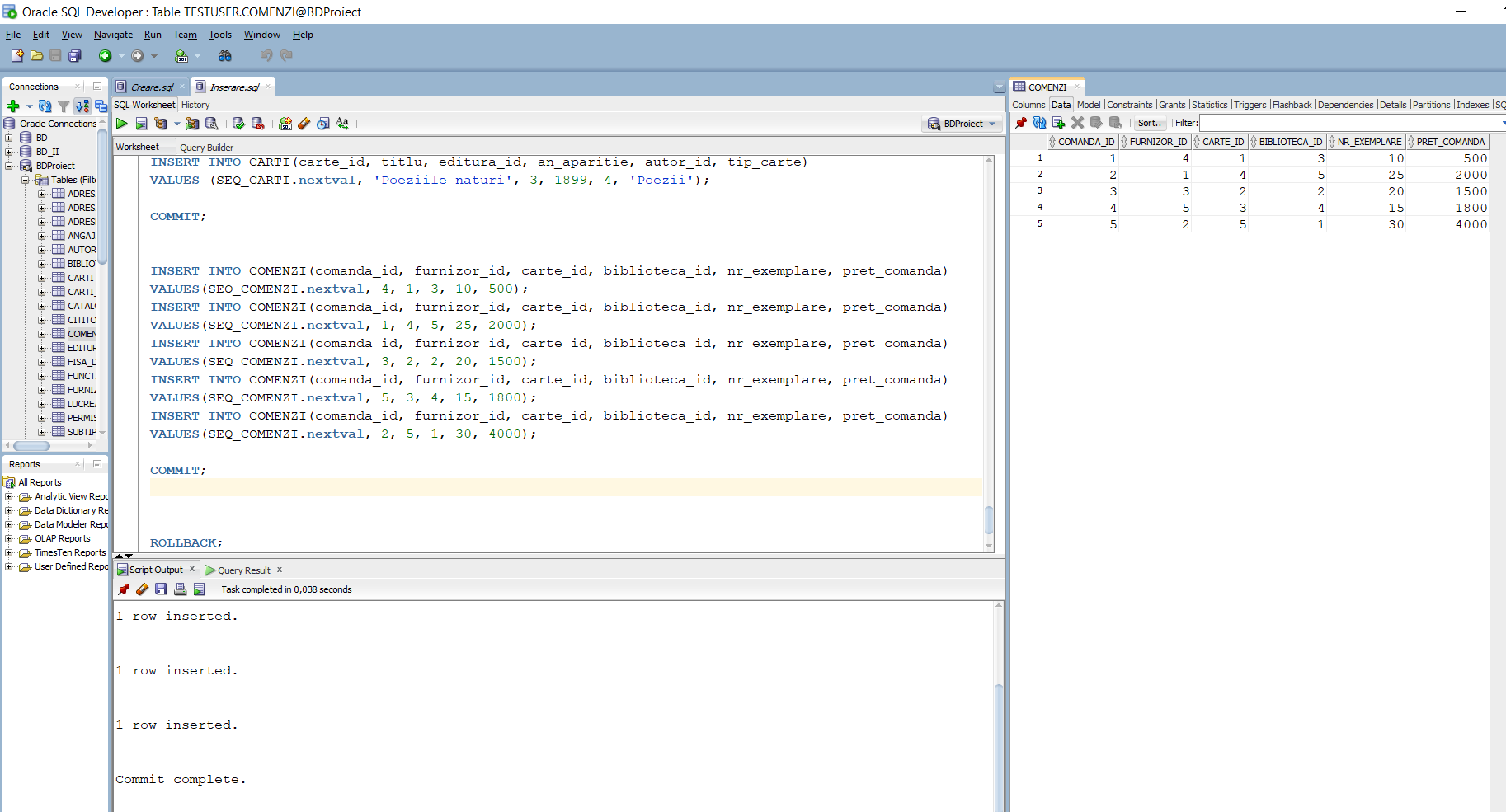
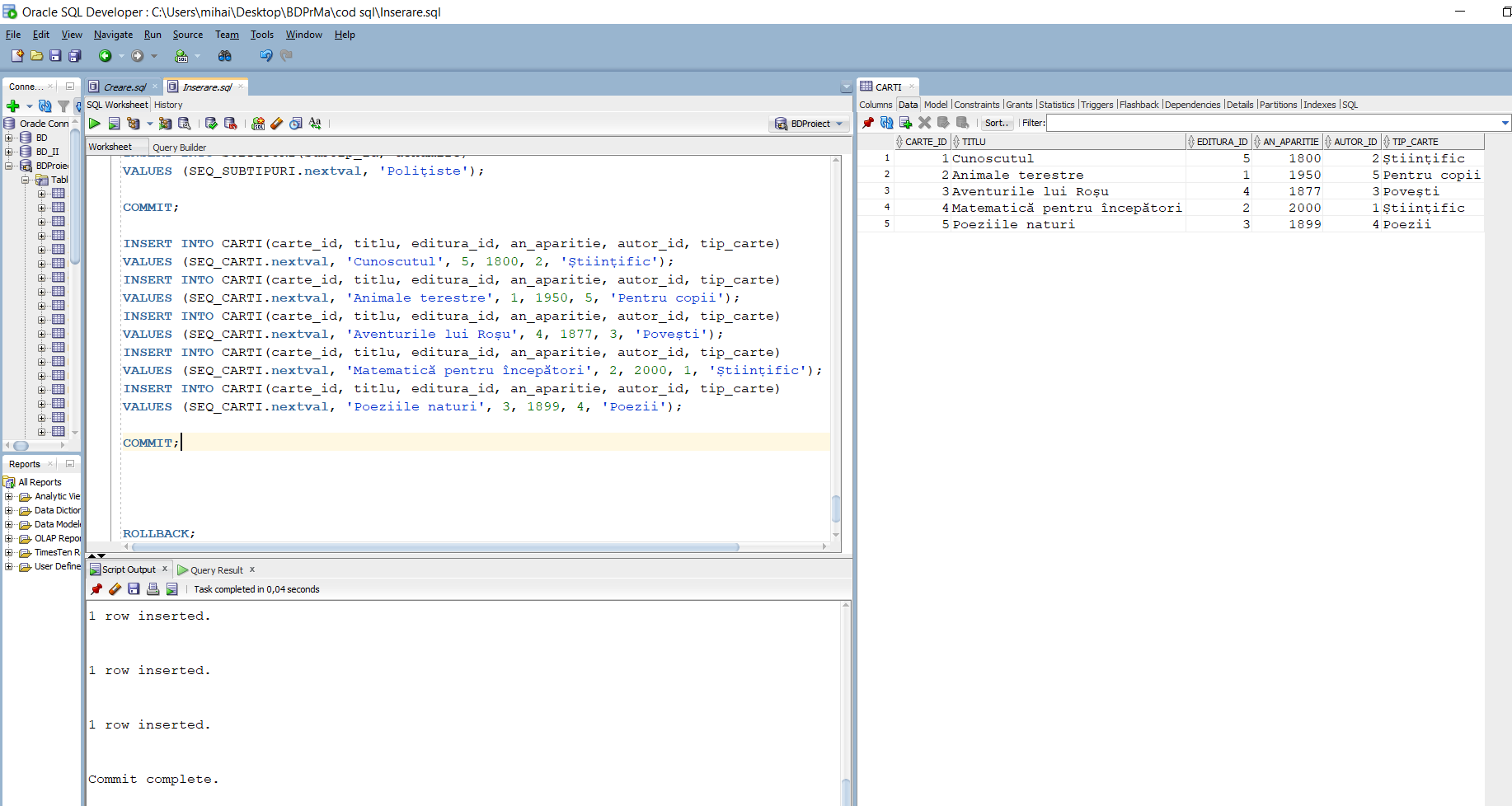
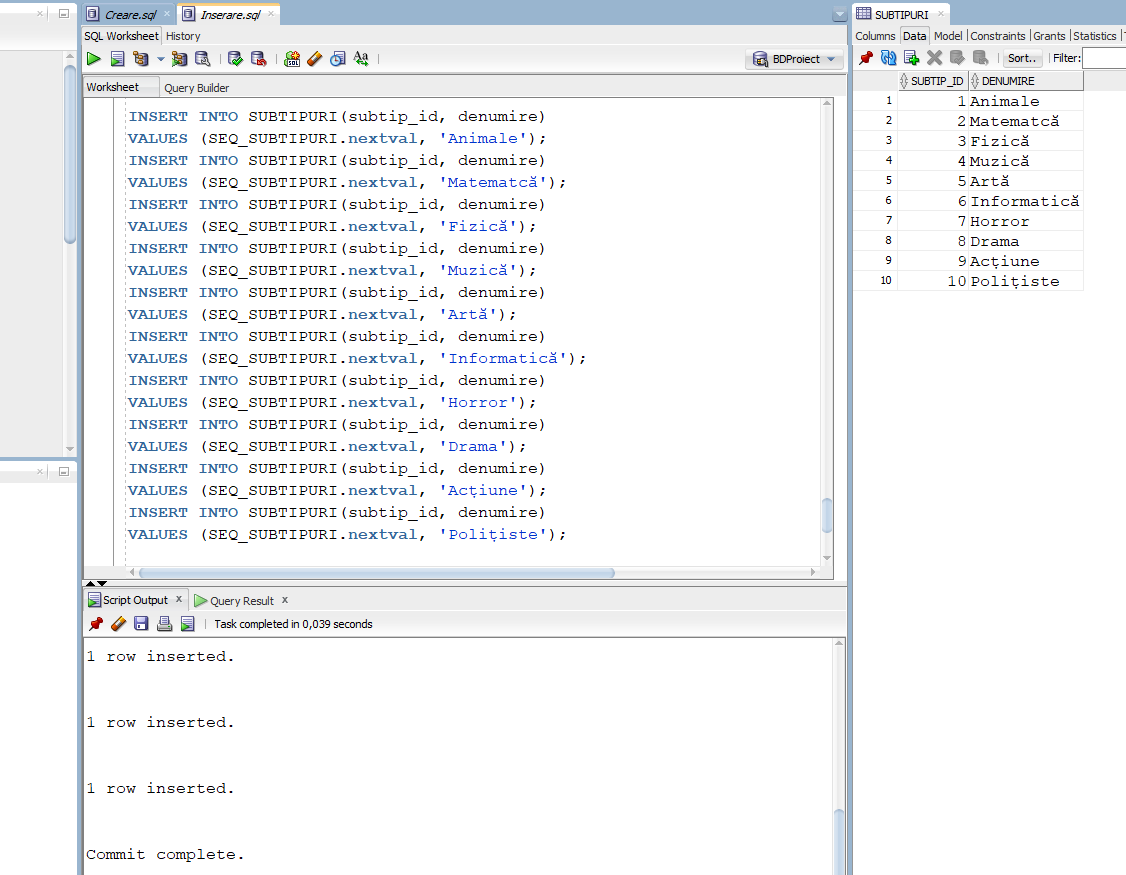
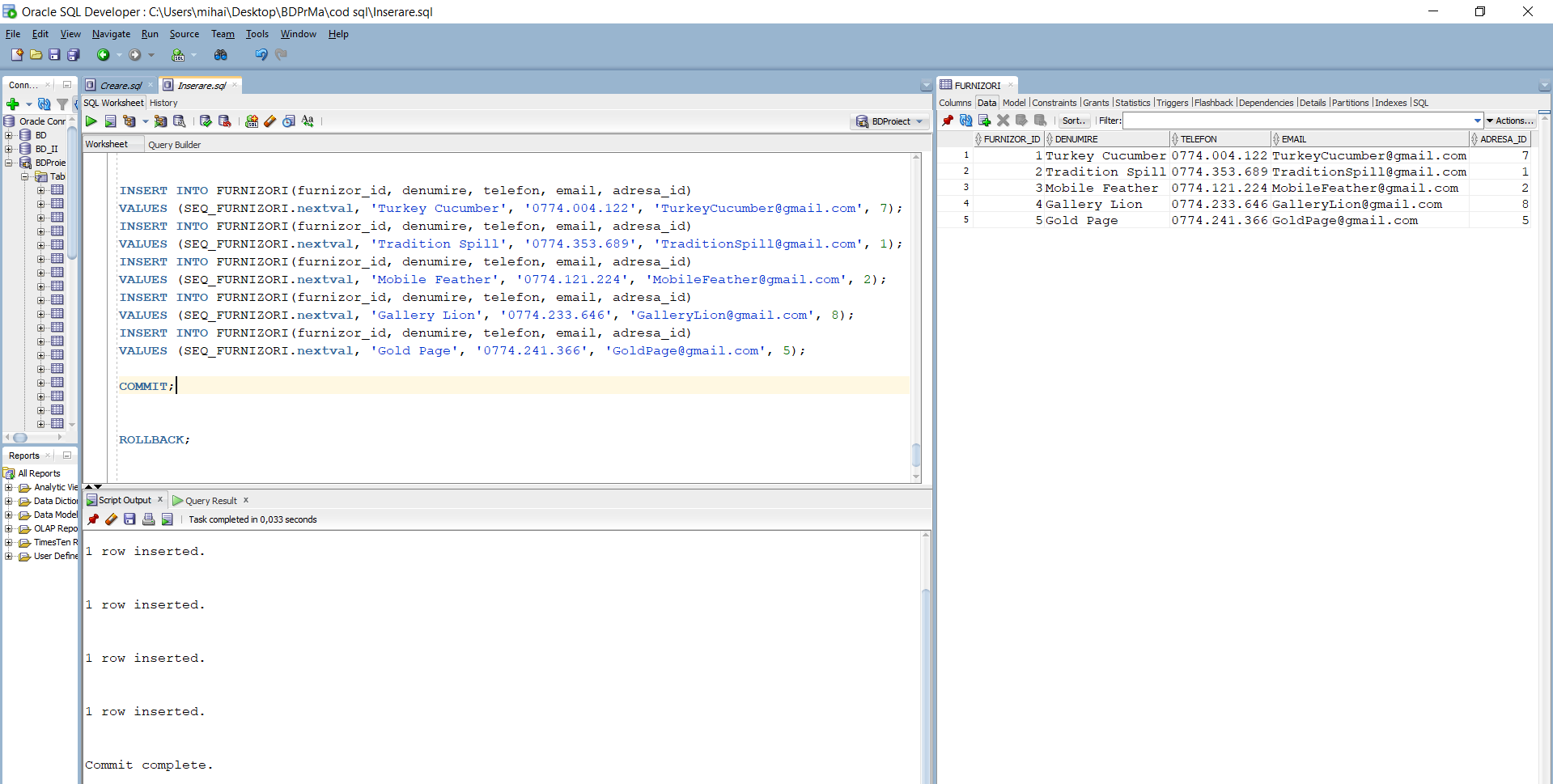
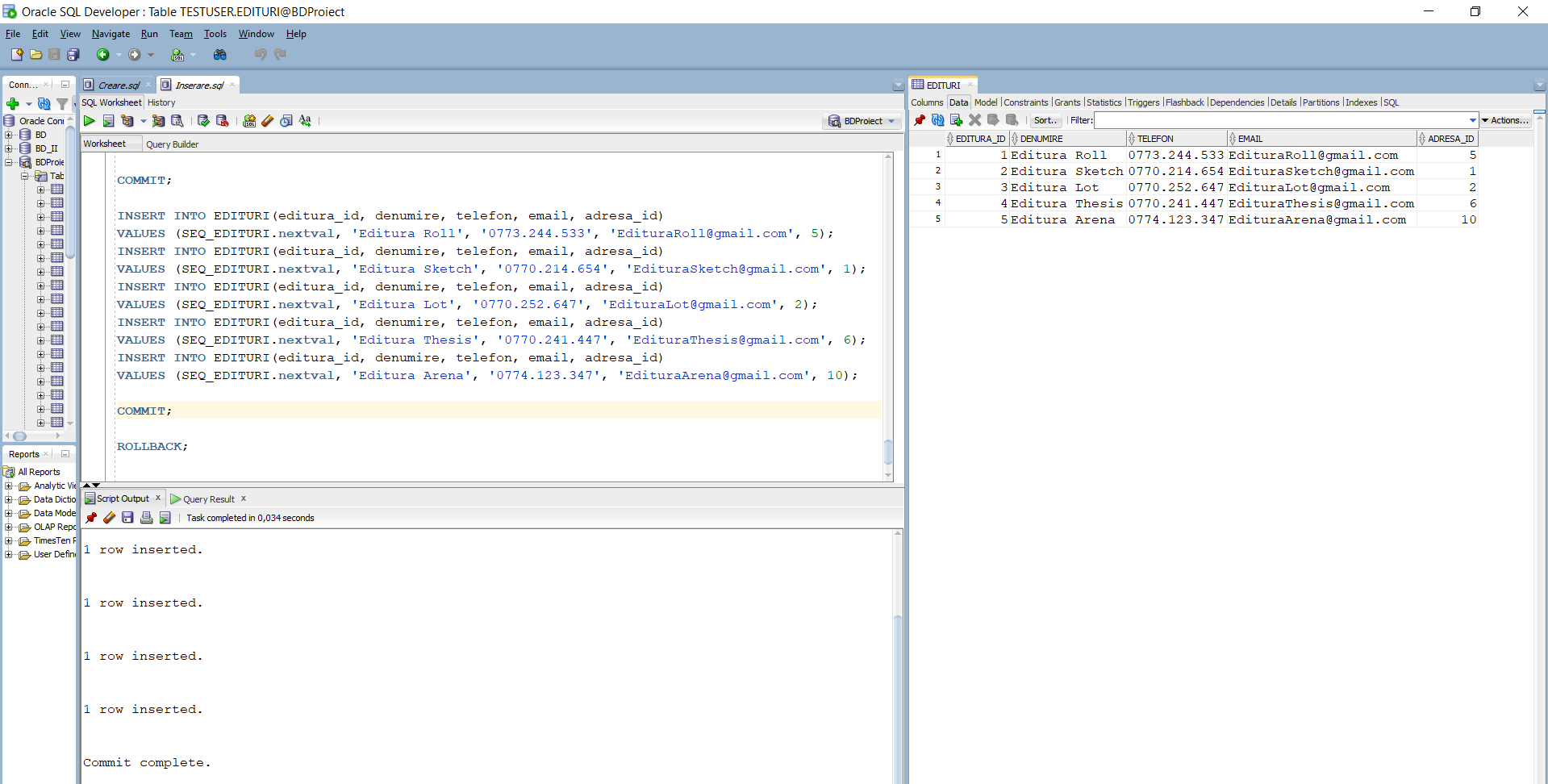
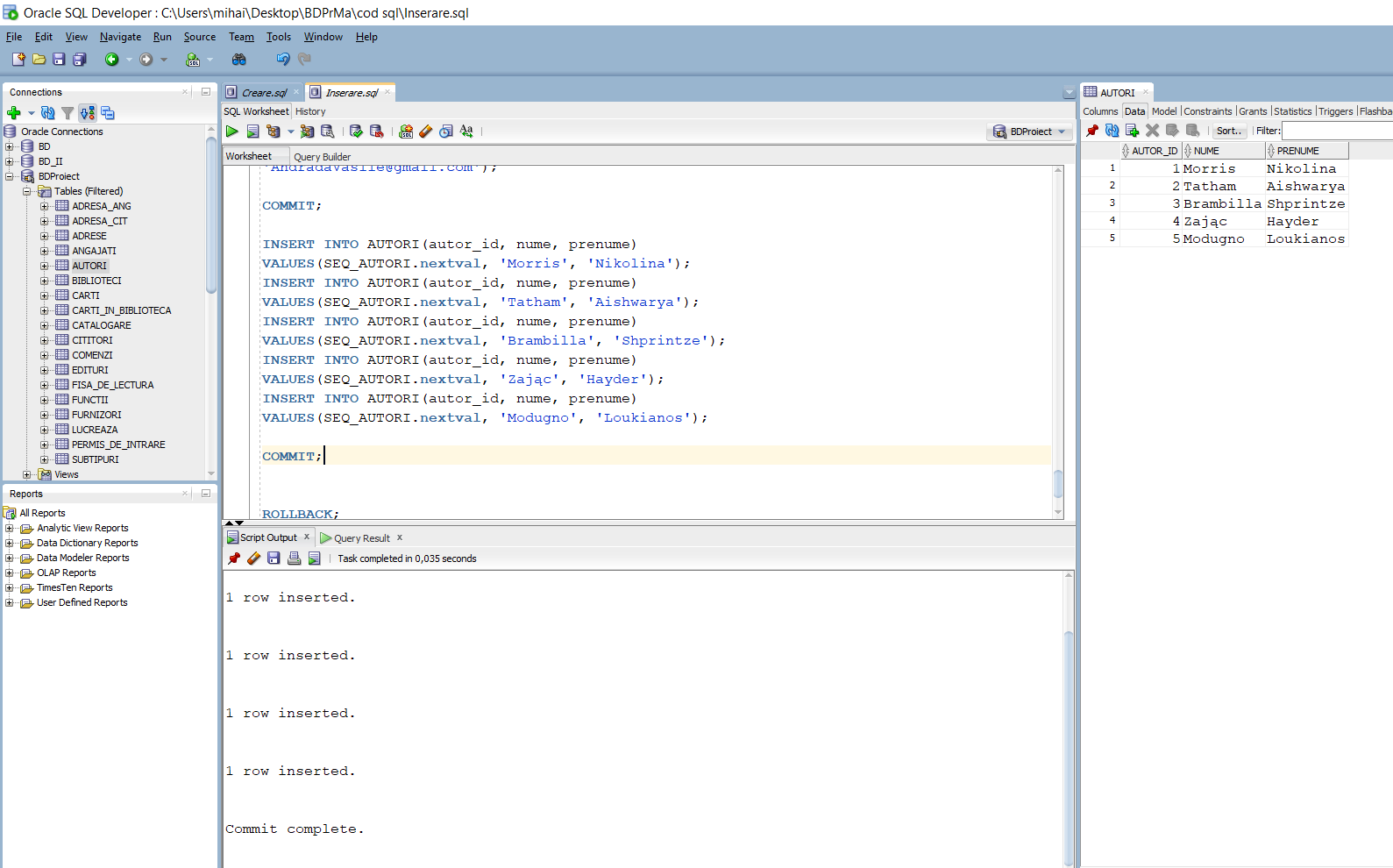
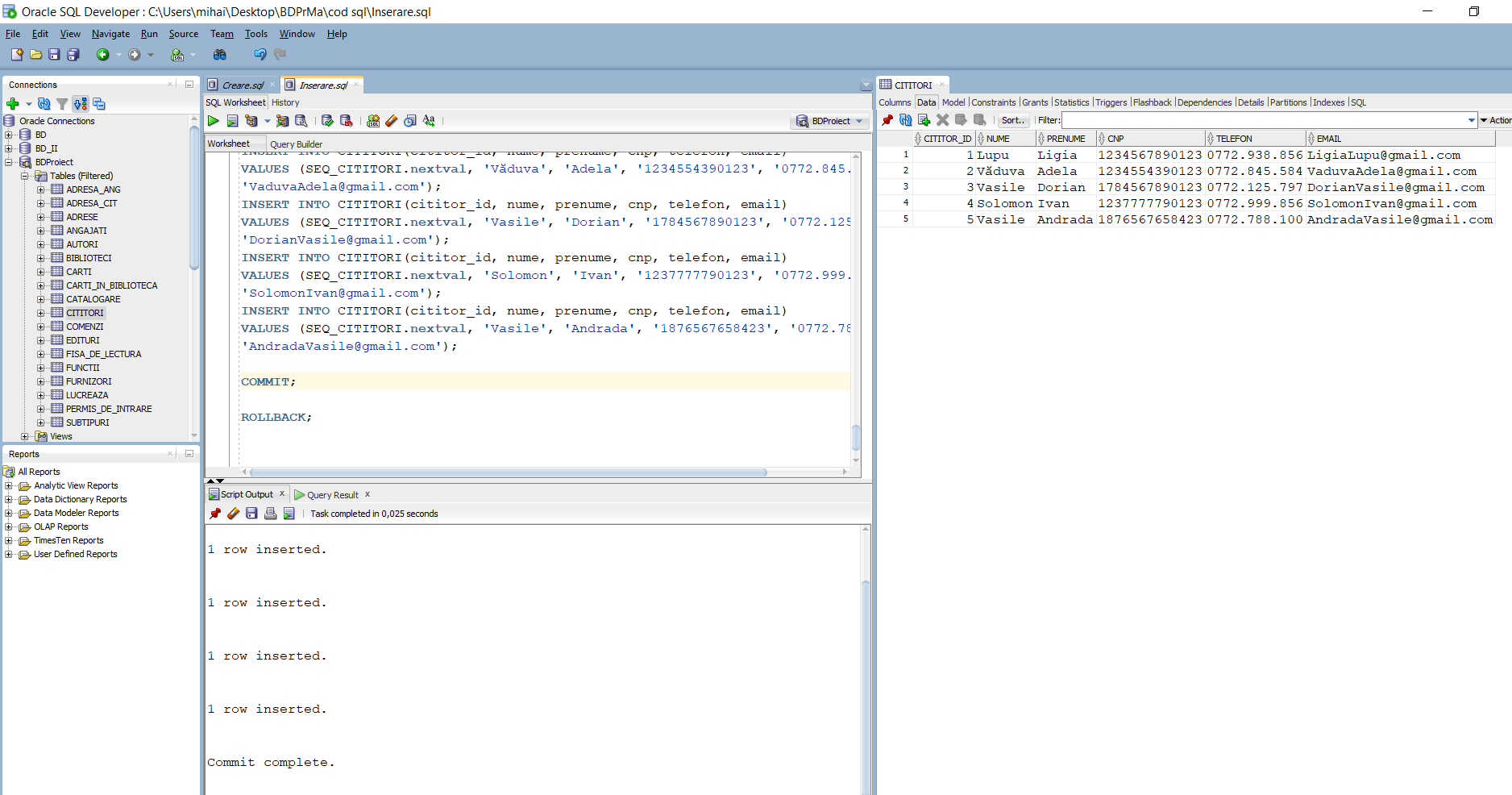
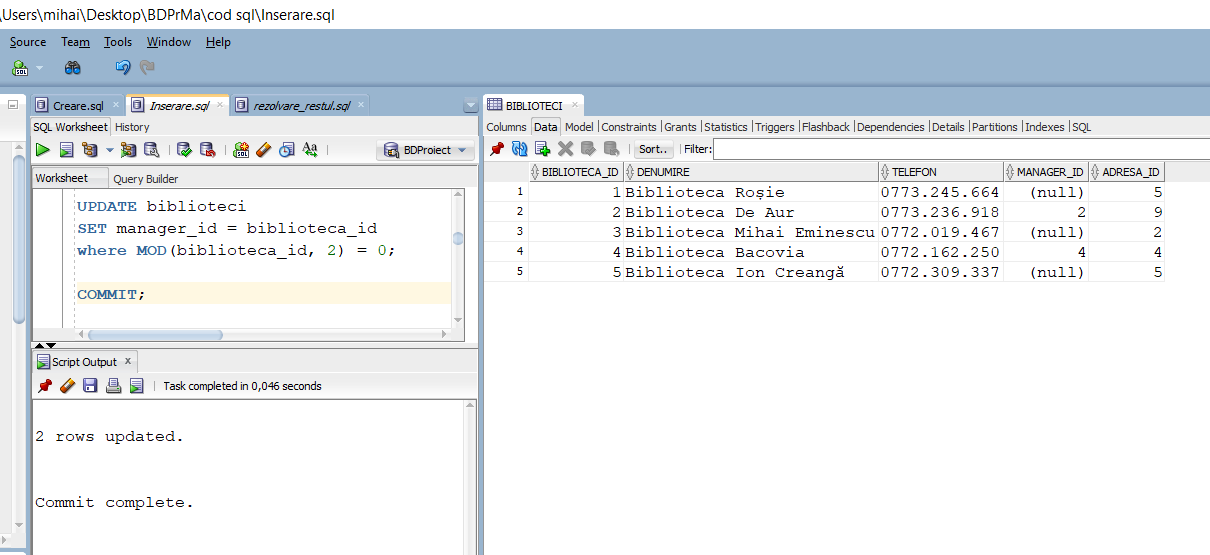
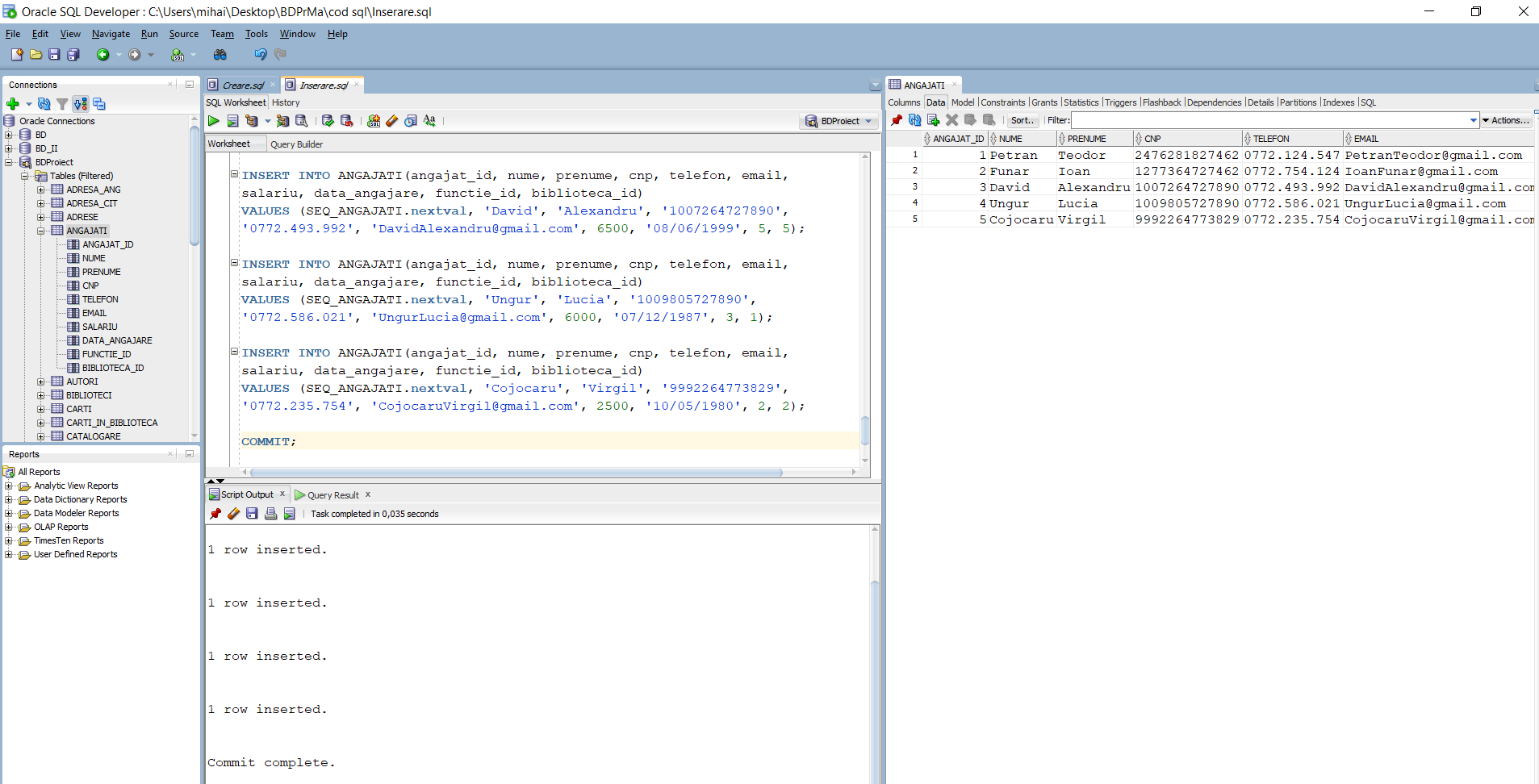
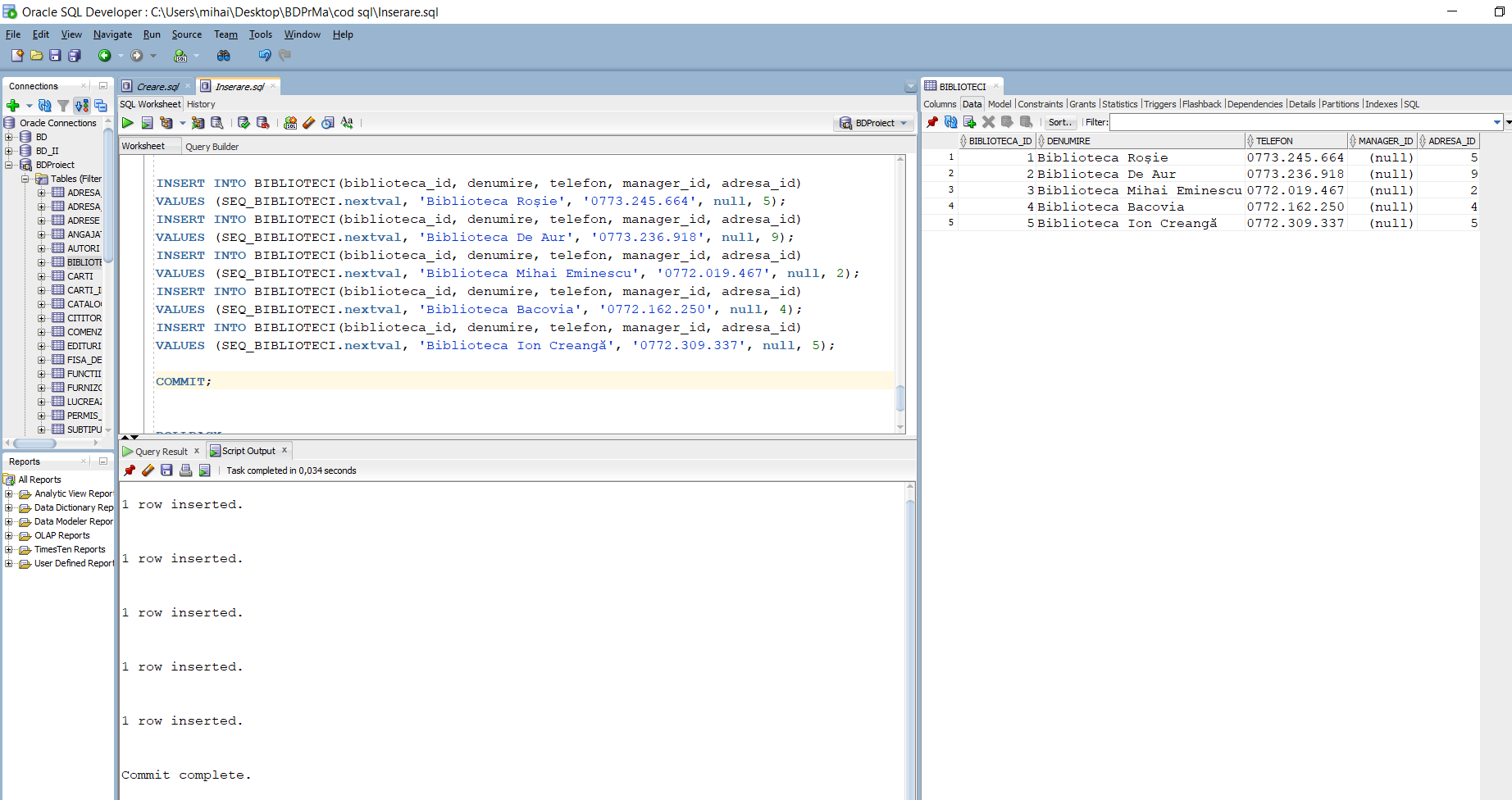
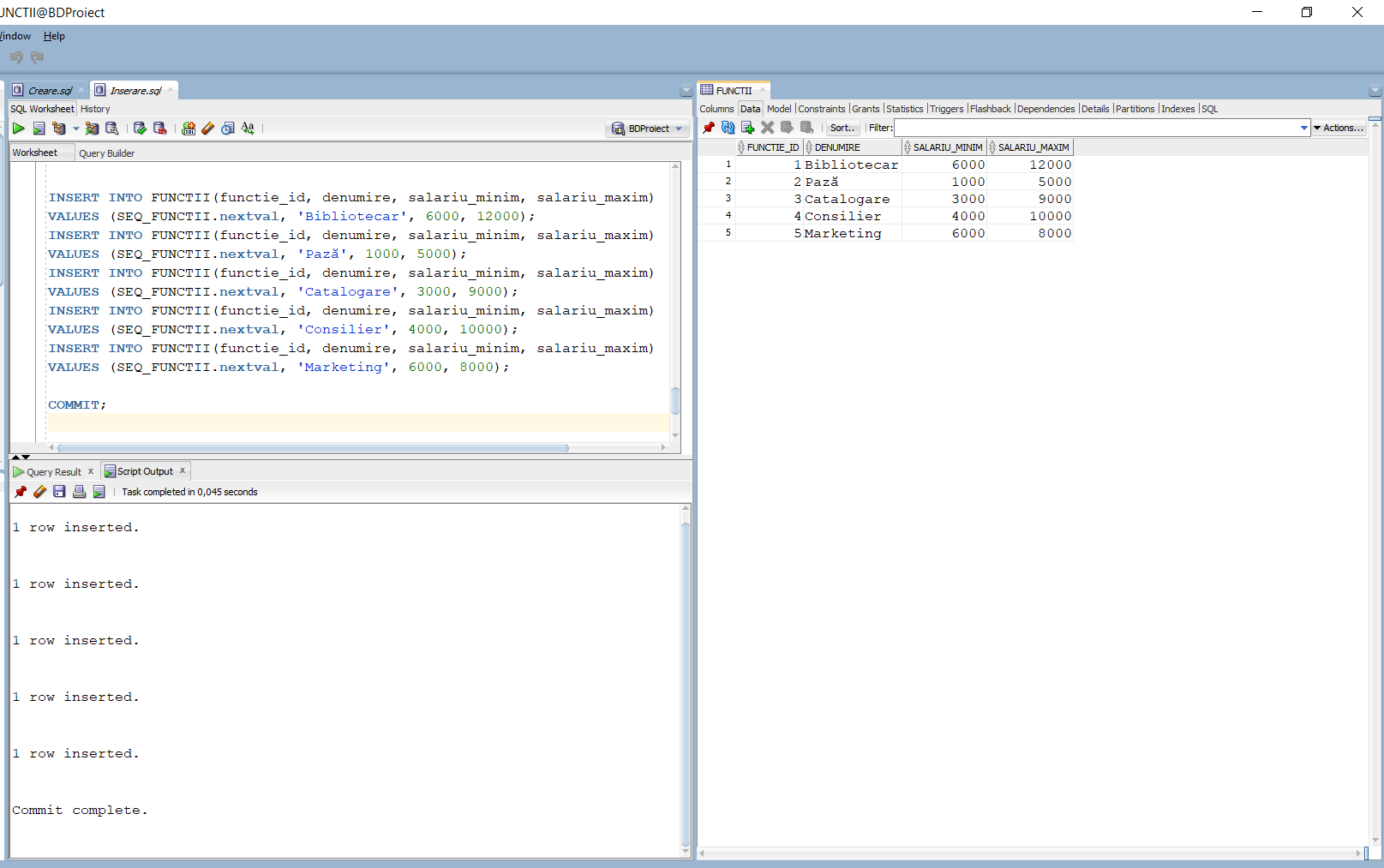
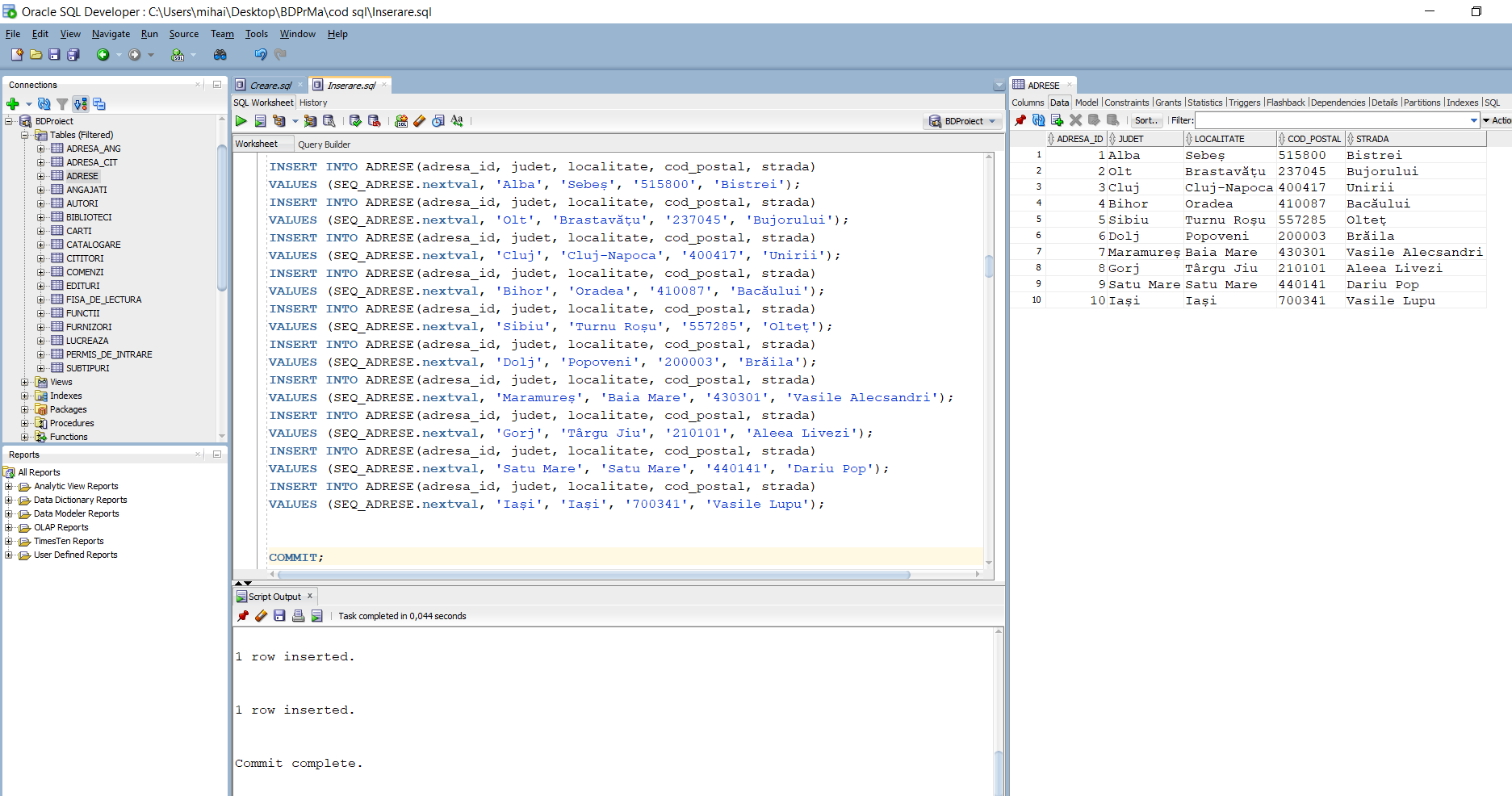
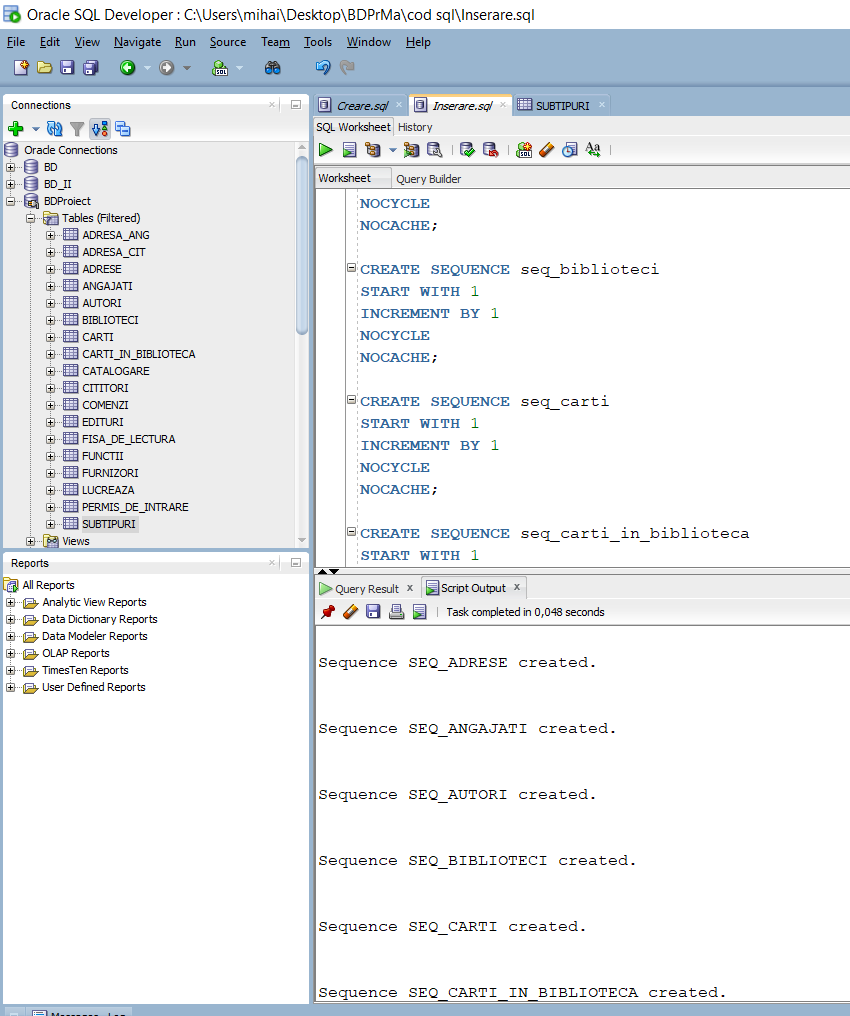
INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(5, 8);

INSERT INTO ADRESA\_CIT(cititor\_id, adresa\_id)

VALUES(5, 2);

COMMIT;



**6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.**

Să se afișeze date despre toți cititori care au citit un număr mai mare sau egal cu media aritmetică a cărților

citite de fiecare cititor și să li se creeze un permis de intrare pentru biblioteca care are cei mai puțini cititori înscriși. În caz că sunt mai multe biblioteci cu același număr minim de cititori să se creeze permisuri doar pentru una dintre ele. Dacă un cititor are deja un permis de intrare la biblioteca respectivă acesta nu va primi unul nou.

create or replace procedure ex6 as

type cititori\_type is table of cititori%rowtype;

type biblioteci\_type is table of biblioteci%rowtype index by pls\_integer;

t\_cititori cititori\_type := cititori\_type();

t\_biblioteci biblioteci\_type;

nr number;

rand\_nr pls\_integer;

v\_cititor cititori%rowtype;

v\_biblioteca biblioteci%rowtype;

v\_an number;

v\_exists number;

begin

select c.\*

bulk collect into t\_cititori

from cititori c, (select c.cititor\_id, count(\*) nr\_carti\_citite

from cititori c, fisa\_de\_lectura fl

where c.cititor\_id = fl.cititor\_id

group by c.cititor\_id

having count(\*) >= (select avg(nr\_carti\_citite)

from (select c.cititor\_id, count(\*) nr\_carti\_citite

from cititori c, fisa\_de\_lectura fl

where c.cititor\_id = fl.cititor\_id

group by c.cititor\_id))) t

where c.cititor\_id = t.cititor\_id

order by 1;

dbms\_output.put\_line('Cititori cautati sunt:');

for i in t\_cititori.first..t\_cititori.last loop

v\_cititor := t\_cititori(i);

dbms\_output.put\_line(v\_cititor.nume || ' ' || v\_cititor.prenume || ' ' || v\_cititor.cnp ||

' ' || v\_cititor.telefon || ' ' || v\_cititor.email);

end loop;

dbms\_output.put\_line('');

select b.\*

bulk collect into t\_biblioteci

from biblioteci b, (select biblioteca\_id, count(\*) nr\_cititori

from permis\_de\_intrare

group by biblioteca\_id

having count(\*) = (select min(count(\*)) nr\_cititori

from permis\_de\_intrare

group by biblioteca\_id)) t

where b.biblioteca\_id = t.biblioteca\_id

order by 1;

nr := t\_biblioteci.last;

select dbms\_random.value(1,nr)

into rand\_nr

from dual;

v\_biblioteca := t\_biblioteci(rand\_nr);

dbms\_output.put\_line('Biblioteca aleasa pentru crearea permiselor este '||v\_biblioteca.denumire);

v\_an := extract(year from sysdate);

for i in t\_cititori.first..t\_cititori.last loop

v\_cititor := t\_cititori(i);

select count(\*)

into v\_exists

from permis\_de\_intrare

where cititor\_id = v\_cititor.cititor\_id

and biblioteca\_id = v\_biblioteca.biblioteca\_id;

if v\_exists != 0 then

continue;

end if;

insert into permis\_de\_intrare(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

values(seq\_permis\_de\_intrare.nextval, v\_an, v\_cititor.cititor\_id, v\_biblioteca.biblioteca\_id);

end loop;

exception

when others then

dbms\_output.put\_line('Ceva nu a mers bine!');

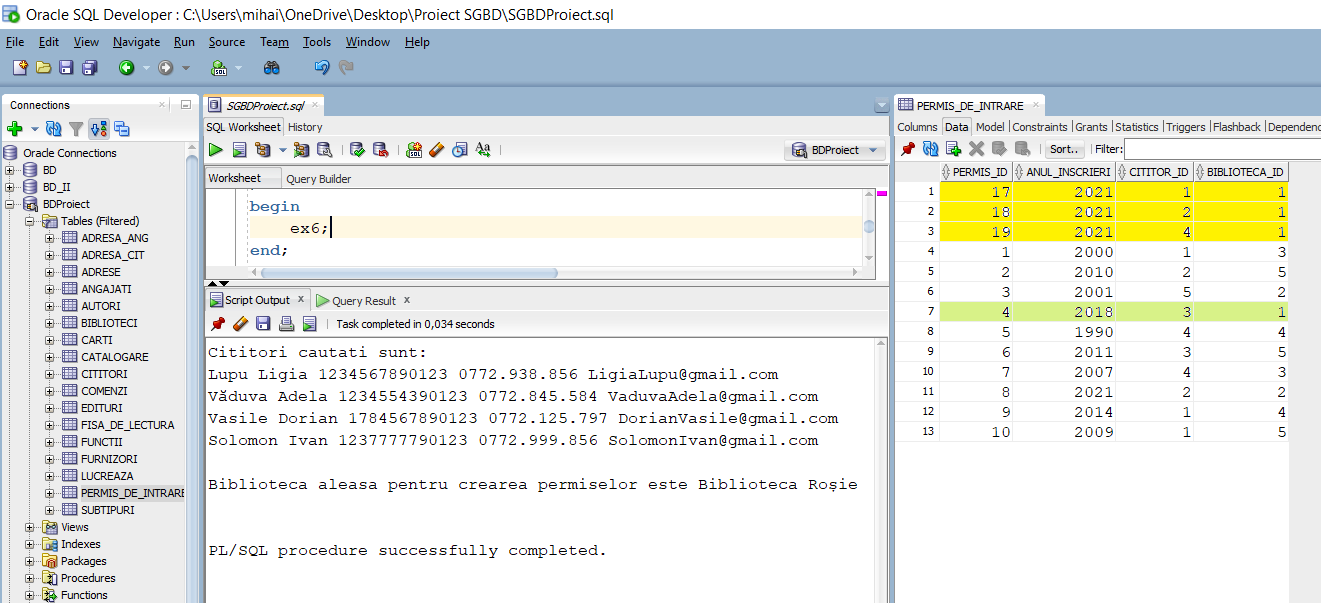
end ex6;

/

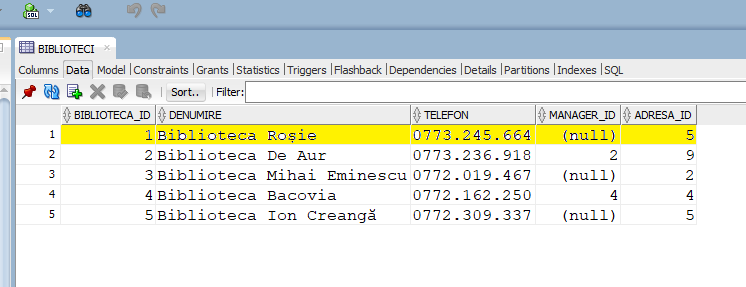
Înaintea executări subprogramului stocat:

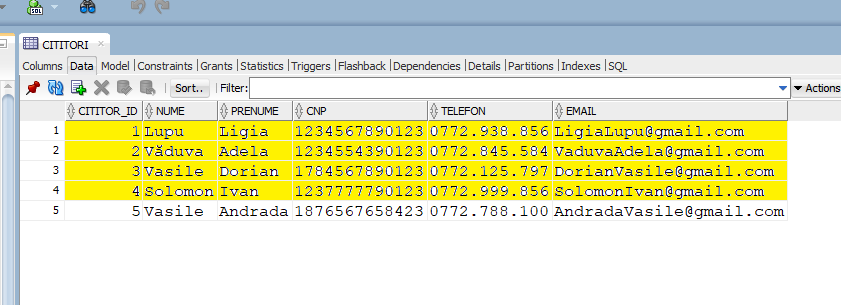


După executarea subprogramului stocat:



Datele din tabelele cititori si biblioteci:





Se observă că cititorul Vasile Dorian avea deja un permis în Biblioteca Roșie, prin urmare nu a fost creat un permis nou pentru acest cititor.

**7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.**

Să li se măreasca salariul cu 10% tuturor angajaților care lucrează în bibliotecile cu cel mai mare număr de cititori înscriși. Dacă salariul angajatului depășește salariul maxim funcției respective, salariul lui va rămâne egal cu salariul maxim. Să se afișeze numărul maxim de cititori înscriși și bibliotecile care au acest număr maxim de cititori înscriși.

create or replace procedure ex7 as

type bibl\_and\_count\_record is record (v\_biblioteca\_id biblioteci.biblioteca\_id%type, v\_count number);

type biblioteci\_and\_count is table of bibl\_and\_count\_record

index by pls\_integer;

cursor c\_angajati (v\_biblioteca\_id angajati.biblioteca\_id%type) is

select \* from angajati where biblioteca\_id = v\_biblioteca\_id;

t\_biblioteci biblioteci\_and\_count;

v\_biblioteca\_id biblioteci.biblioteca\_id%type;

v\_functie\_id functii.functie\_id%type;

max\_sal functii.salariu\_maxim%type;

old\_sal angajati.salariu%type;

new\_sal angajati.salariu%type;

v\_denumire biblioteci.denumire%type;

begin

select biblioteca\_id, count(cititor\_id)

bulk collect into t\_biblioteci

from permis\_de\_intrare

group by biblioteca\_id

having count(cititor\_id) = (select max(count(cititor\_id))

from permis\_de\_intrare

group by biblioteca\_id);

dbms\_output.put\_line('Numarul maxim de cititori inscrisi intr-o biblioteca este '||t\_biblioteci(1).v\_count);

dbms\_output.put\_line('');

dbms\_output.put\_line('Iar aceste biblioteci sunt:');

for i in t\_biblioteci.first..t\_biblioteci.last loop

v\_biblioteca\_id := t\_biblioteci(i).v\_biblioteca\_id;

select denumire into v\_denumire from biblioteci where biblioteca\_id = v\_biblioteca\_id;

dbms\_output.put\_line(v\_denumire);

for v\_ang in c\_angajati(v\_biblioteca\_id) loop

select functie\_id, salariu

into v\_functie\_id, old\_sal

from angajati

where angajat\_id = v\_ang.angajat\_id;

select salariu\_maxim

into max\_sal

from functii

where functie\_id = v\_functie\_id;

new\_sal := old\_sal \* 1.1;

if new\_sal > max\_sal then

new\_sal := max\_sal;

end if;

update angajati

set salariu = new\_sal

where angajat\_id = v\_ang.angajat\_id;

end loop;

end loop;

exception

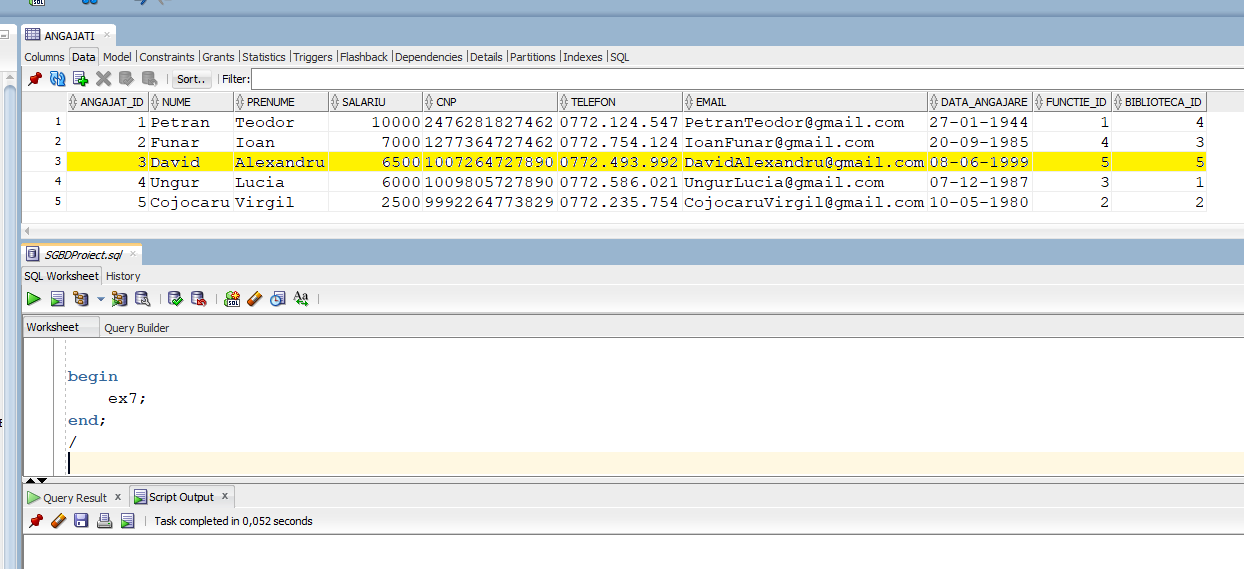
when others then

dbms\_output.put\_line('Ceva nu a mers bine!');

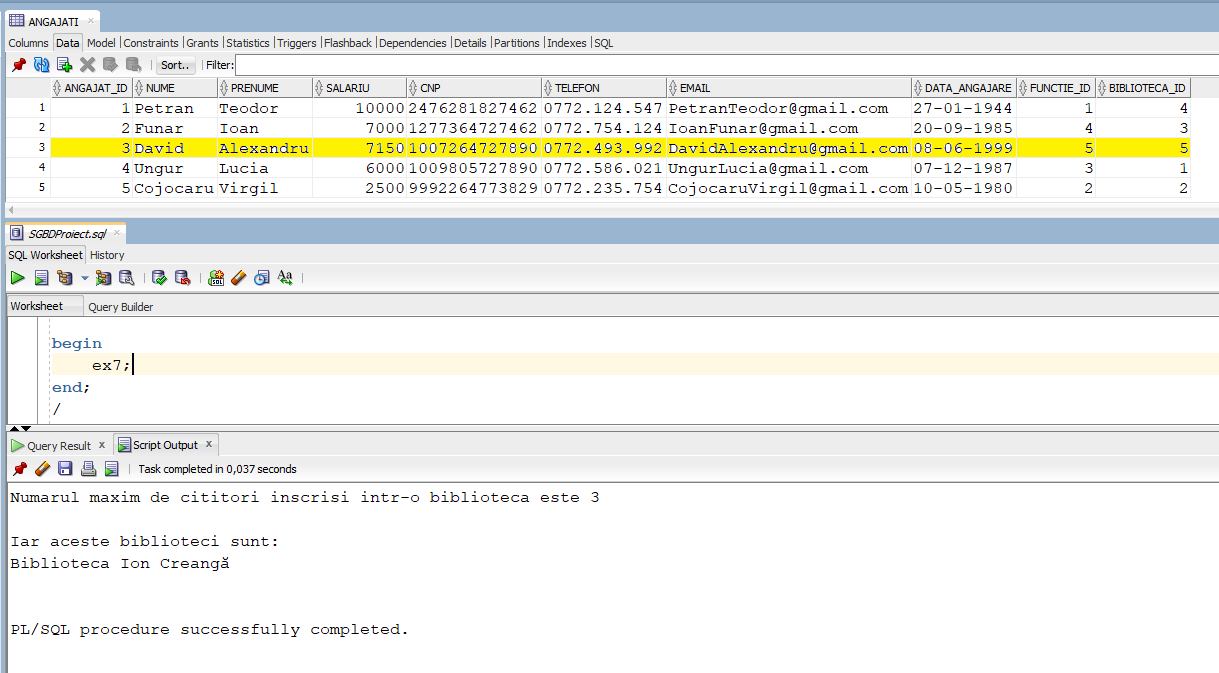
end ex7;

/

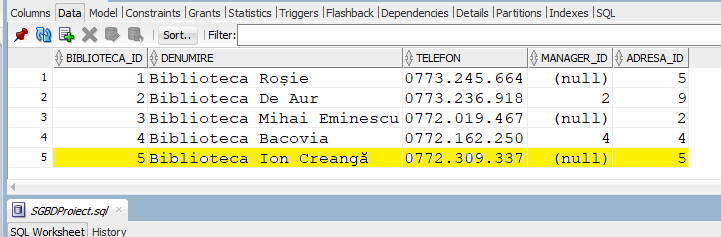
Înaintea executări subprogramului stocat:



După executări subprogramului stocat:



Datele din tabelul biblioteci:



**8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

Scrieți o funcție care primește ca parametru un șir de caractere și returnează numărul de exemplare pentru cărțile care conțin șirul dat ca parametru. Afișează pe ecran titlul și autorul cărților respective.

create or replace function ex8 (sir varchar2) return number as

v\_carte\_id carti.carte\_id%type;

v\_res number;

v\_sir varchar2(200);

begin

v\_sir := '%'||lower(sir)||'%';

select carte\_id

into v\_carte\_id

from carti

where lower(titlu) like v\_sir;

for carte in (select ca.titlu, a.nume, a.prenume, sum(co.nr\_exemplare) nr\_exemplare

from carti ca, comenzi co, autori a

where ca.autor\_id = a.autor\_id

and ca.carte\_id = co.carte\_id

and ca.carte\_id = v\_carte\_id

group by ca.titlu, a.nume, a.prenume) loop

v\_res := carte.nr\_exemplare;

dbms\_output.put\_line('Cartea care contine subsirul cautat este: ' || carte.titlu ||' Autorul: ' || carte.nume || ' ' || carte.prenume);

end loop;

return v\_res;

exception

when no\_data\_found then

dbms\_output.put\_line('Nu exista carti care contin subsirul cautat');

return 0;

when too\_many\_rows then

v\_res := 0;

dbms\_output.put\_line('Cartile care contin subsirul cautat sunt:');

dbms\_output.put\_line('');

for carte in (select ca.titlu, a.nume, a.prenume, sum(co.nr\_exemplare) nr\_exemplare

from carti ca, comenzi co, autori a

where ca.autor\_id = a.autor\_id

and ca.carte\_id = co.carte\_id

and lower(ca.titlu) like v\_sir

group by ca.titlu, a.nume, a.prenume) loop

v\_res := v\_res + carte.nr\_exemplare;

dbms\_output.put\_line(carte.titlu ||' Autorul: ' || carte.nume || ' ' || carte.prenume);

end loop;

return v\_res;

when others then

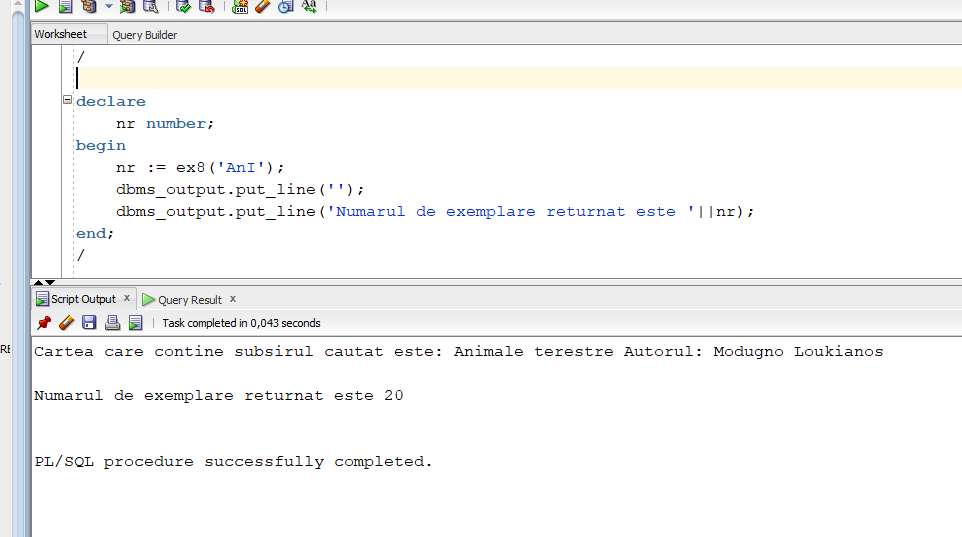
dbms\_output.put\_line('Ceva nu a mers bine');

return -1;

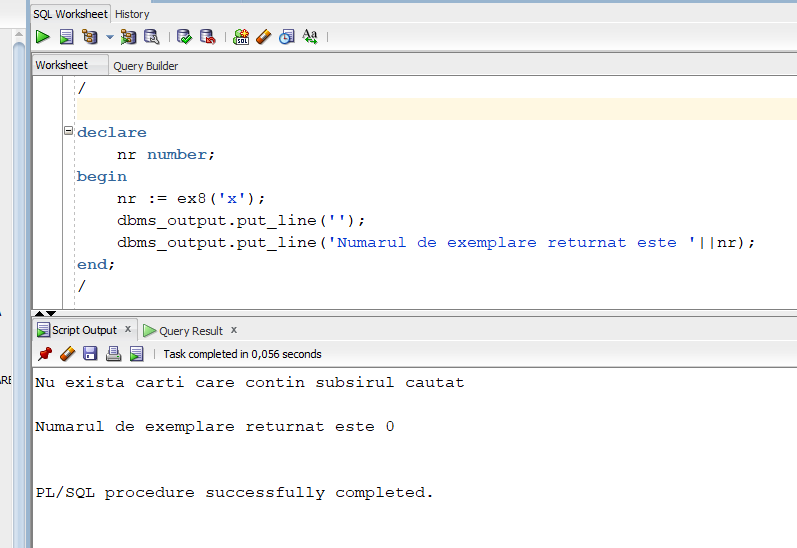
end;

/

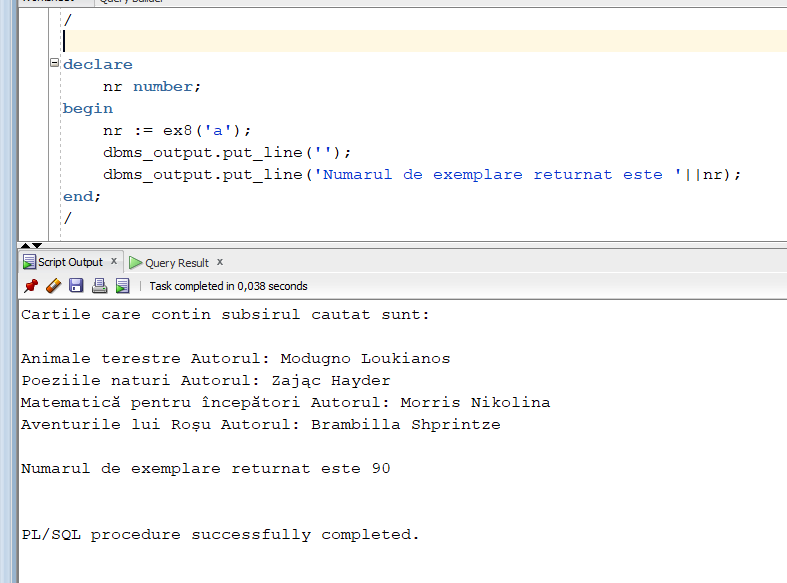
Cazul când există o singură carte cu subșirul dat:



Cazul no\_data\_found:



Cazul too\_many\_rows:



**9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

Scrieți o procedură care primește ca parametru un șir de caractere și pentru cărțile care conțin șirul de caractere

în titlul lor să se afișeze: titlul cărți, autorul cărți, biblioteca la care a fost citită cartea și de cine a fost citită cartea.

create or replace procedure ex9 (sir varchar2) as

v\_carte\_id carti.carte\_id%type;

v\_sir varchar2(200);

begin

v\_sir := '%'||lower(sir)||'%';

select carte\_id

into v\_carte\_id

from carti

where lower(titlu) like v\_sir;

dbms\_output.put\_line('Istoricul carti cautate este: ');

dbms\_output.put\_line('');

for carte in (select distinct ca.titlu, a.nume nume\_autor, a.prenume prenume\_autor,

b.denumire biblioteca, ci.nume cititor\_nume, ci.prenume cititor\_prenume

from carti ca, fisa\_de\_lectura fl, cititori ci, biblioteci b, autori a

where a.autor\_id = ca.autor\_id

and ca.carte\_id = fl.carte\_id

and fl.cititor\_id = ci.cititor\_id

and fl.biblioteca\_id = b.biblioteca\_id

and ca.carte\_id = v\_carte\_id

order by 1) loop

dbms\_output.put\_line(carte.titlu || ' scrisa de ' || carte.nume\_autor || ' ' || carte.prenume\_autor ||

' citita de ' || carte.cititor\_nume || ' ' || carte.cititor\_prenume || ' la biblioteca ' || carte.biblioteca);

end loop;

exception

when no\_data\_found then

dbms\_output.put\_line('Nu exista carti care contin subsirul cautat');

when too\_many\_rows then

dbms\_output.put\_line('Istoricul cartilor cautate este:');

dbms\_output.put\_line('');

for carte in (select distinct ca.titlu, a.nume nume\_autor, a.prenume prenume\_autor,

b.denumire biblioteca, ci.nume cititor\_nume, ci.prenume cititor\_prenume

from carti ca, fisa\_de\_lectura fl, cititori ci, biblioteci b, autori a

where a.autor\_id = ca.autor\_id

and ca.carte\_id = fl.carte\_id

and fl.cititor\_id = ci.cititor\_id

and fl.biblioteca\_id = b.biblioteca\_id

and lower(ca.titlu) like v\_sir

order by 1) loop

dbms\_output.put\_line(carte.titlu || ' scrisa de ' || carte.nume\_autor || ' ' || carte.prenume\_autor ||

' citita de ' || carte.cititor\_nume || ' ' || carte.cititor\_prenume || ' la biblioteca ' || carte.biblioteca);

end loop;

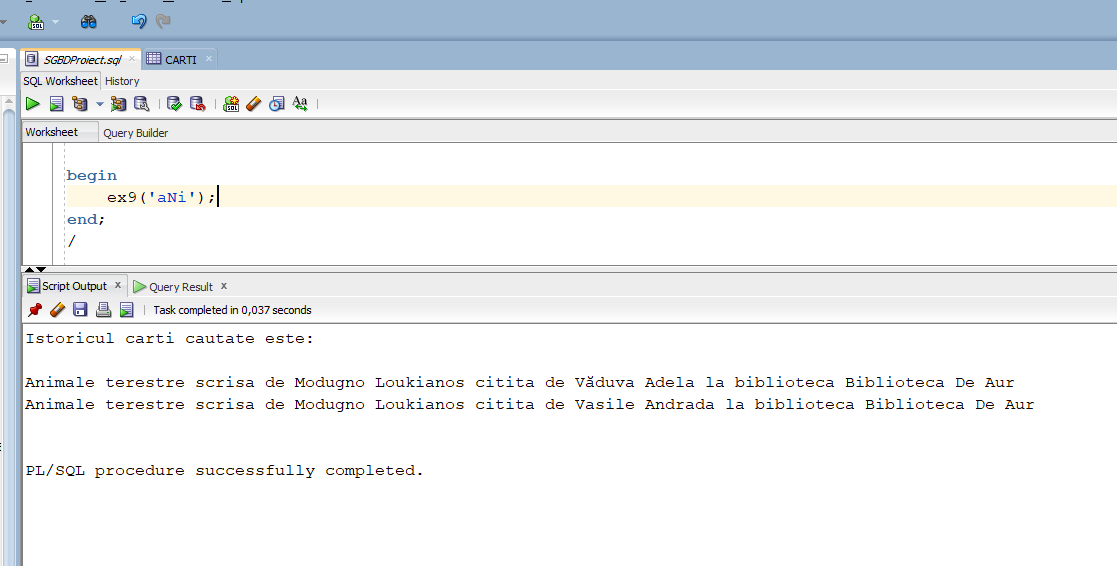
when others then

dbms\_output.put\_line('Ceva nu a mers bine');

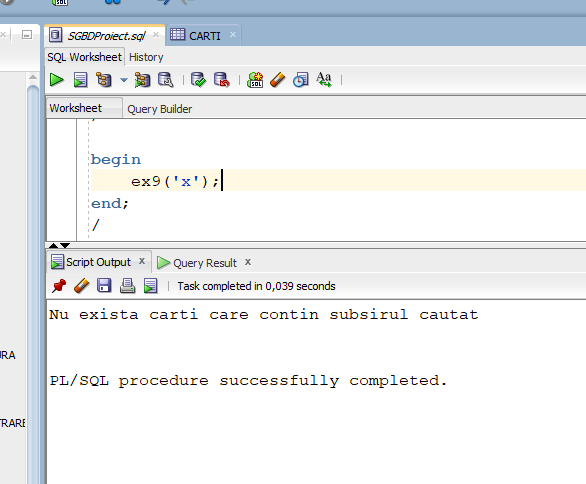
end;

/

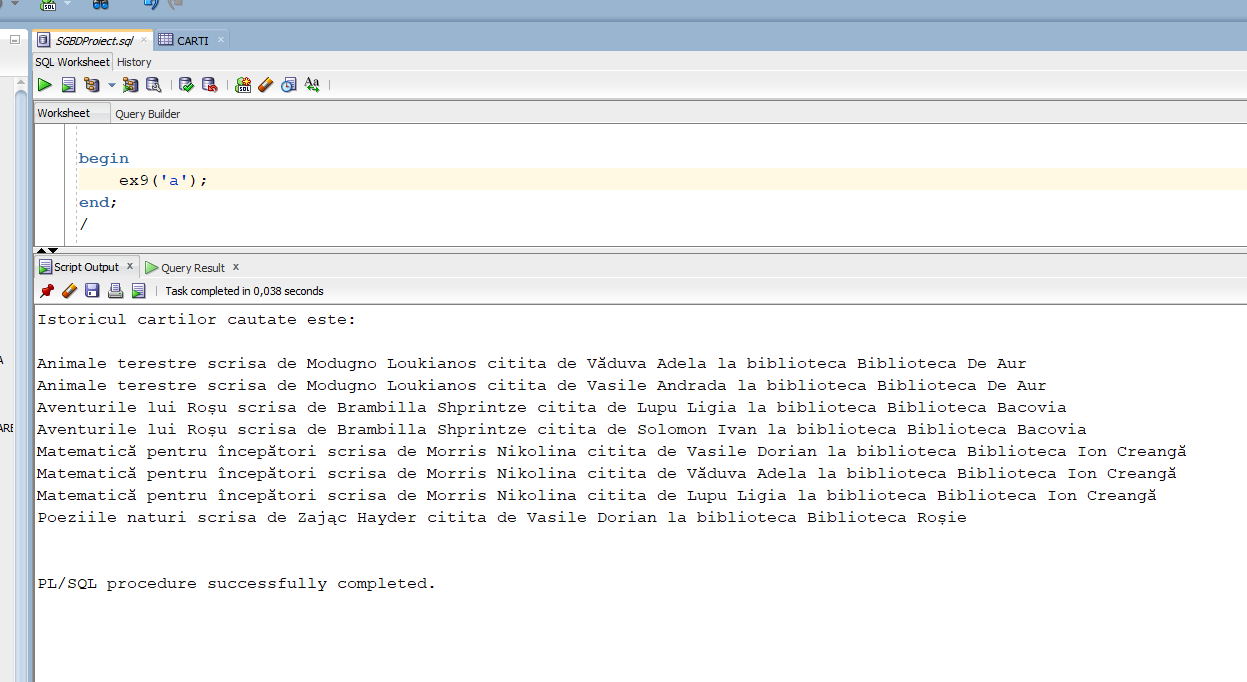
Cazul când există o singură carte cu subșirul dat:



Cazul no\_data\_found:



Cazul too\_many\_rows:



**10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.**

Definiți un trigger de tip LMD la nivel de comandă care permite inserarea, actualizarea și ștergerea datelor

din tabelul fisa\_de\_lectrura doar în intervalul orar 9-20 și care contorizează numărul de acțiuni insert, update și delete făcute.

create or replace package contor\_iud as

nr\_insert number := 0;

nr\_update number := 0;

nr\_delete number := 0;

end contor\_iud;

/

create or replace trigger trig\_fisa

before insert or update or delete on fisa\_de\_lectura

begin

if to\_char(sysdate, 'HH24') not between 9 and 20 then

raise\_application\_error(-20001, 'Operatile asupra tabelului sunt permise doar in intervalul 9-20');

else

if updating then

contor\_iud.nr\_update := contor\_iud.nr\_update + 1;

elsif inserting then

contor\_iud.nr\_insert := contor\_iud.nr\_insert + 1;

else

contor\_iud.nr\_delete := contor\_iud.nr\_delete + 1;

end if;

end if;

end;

/

**Dupa executarea instrucțiunilor următoare in intervalul 9-20 avem:**

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(11, 1, 1, 3, 1, '11/02/2000', '25/02/2000');

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(12, 4, 3, 1, 2, '01/04/2005', '15/04/2005');

update fisa\_de\_lectura

set data\_restituire = sysdate

where fisa\_id > 10;

delete from fisa\_de\_lectura

where fisa\_id = 5;

delete from fisa\_de\_lectura

where fisa\_id = 1;

delete from fisa\_de\_lectura

where fisa\_id = 11;

begin

dbms\_output.put\_line('Asupra tabelului fisa\_de\_lectura s-au efectuat:');

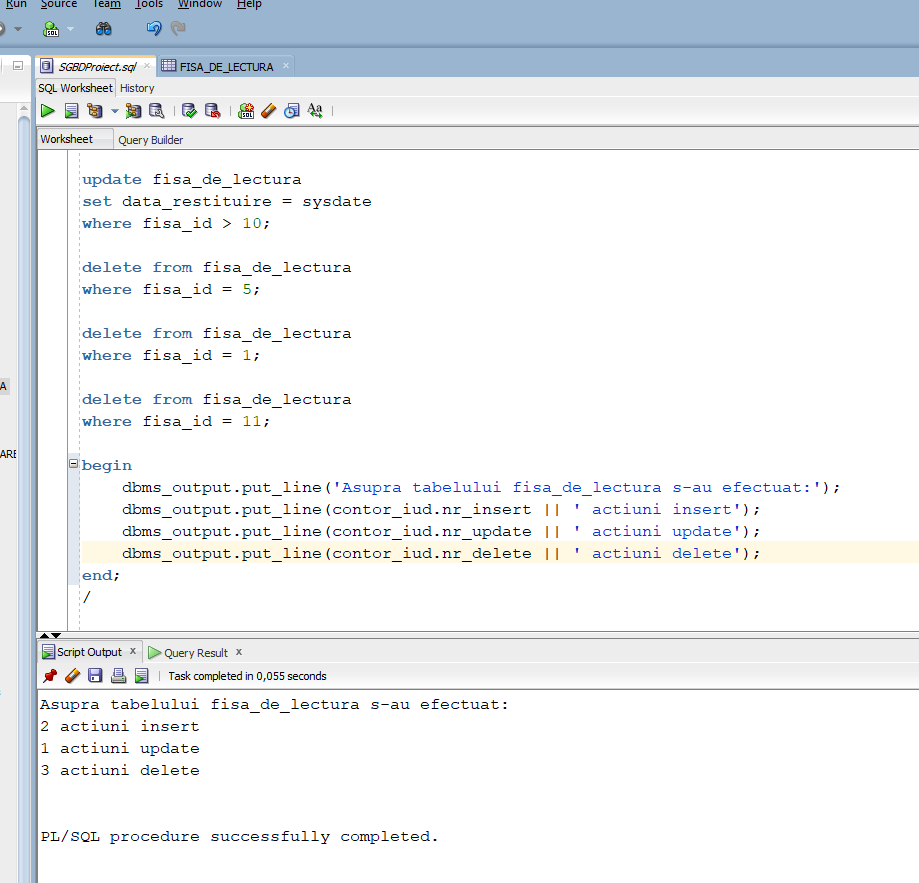
dbms\_output.put\_line(contor\_iud.nr\_insert || ' actiuni insert');

dbms\_output.put\_line(contor\_iud.nr\_update || ' actiuni update');

dbms\_output.put\_line(contor\_iud.nr\_delete || ' actiuni delete');

end;

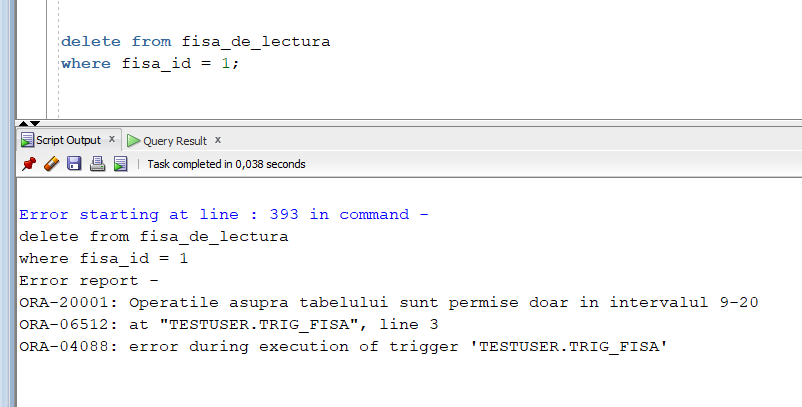
/



**Dupa executarea instrucțiuni următoare in afara intervalului 9-20 avem:**

delete from fisa\_de\_lectura

where fisa\_id = 1;



**11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.**

Definiți un trigger LMD la nivel de linie care verifică corectitudinea datelor introduse în tabelul fisa\_de\_lectura.

create or replace trigger trig\_fisa2

before insert or update on fisa\_de\_lectura

for each row

declare

v\_permis\_exists number;

v\_carte\_exists number;

begin

v\_permis\_exists := 0;

v\_carte\_exists := 0;

select count(\*)

into v\_permis\_exists

from permis\_de\_intrare

where permis\_id = :new.permis\_id

and cititor\_id = :new.cititor\_id

and biblioteca\_id = :new.biblioteca\_id;

select count(\*)

into v\_carte\_exists

from comenzi

where carte\_id = :new.carte\_id

and biblioteca\_id = :new.biblioteca\_id;

if v\_permis\_exists = 0 or v\_carte\_exists = 0 then

raise\_application\_error(-20001, 'Datele au fost introduse gresit');

end if;

end;

/

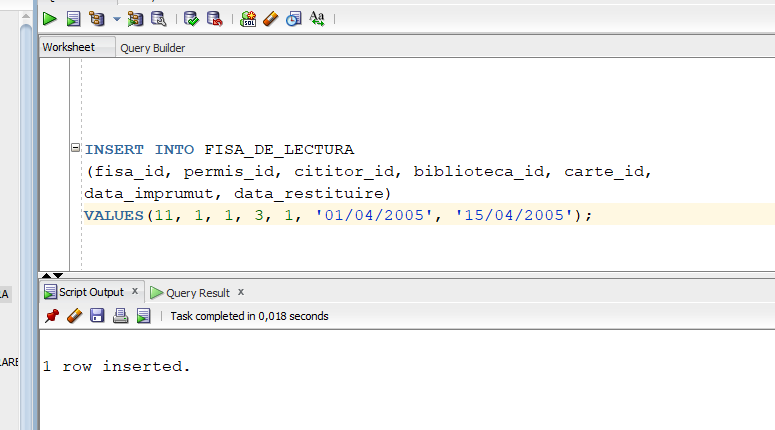
Dacă datele au fost introduse corect avem:

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(11, 1, 1, 3, 1, '01/04/2005', '15/04/2005');



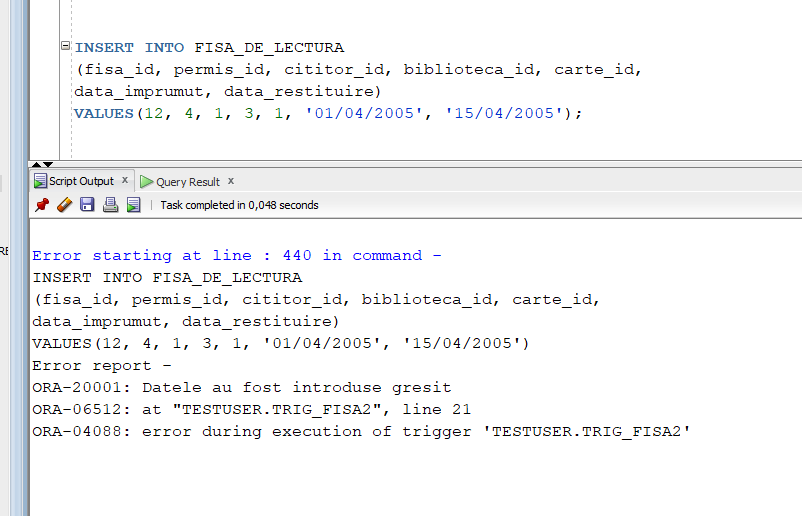
Dacă datele au fost introduse incorect avem:

INSERT INTO FISA\_DE\_LECTURA

(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

VALUES(12, 4, 1, 3, 1, '01/04/2005', '15/04/2005');



**12. Definiți un trigger de tip LDD. Declanșați trigger-ul.**

Definiți un trigger de tip LDD care interzice acțiunile de tip LDD în a doua jumătate a zilei

și memorează în tabelul istoric\_LDD istoricul acțiunilor de tip LDD realizate.

create table istoric\_LDD(

username varchar2(50),

nume\_bd varchar2(50),

eveniment varchar2(50),

nume\_obiect varchar2(50),

data TIMESTAMP

);

create or replace trigger trig\_LDD

before create or alter or drop on schema

begin

if to\_char(sysdate, 'HH24') >= 12 then

raise\_application\_error(-20001, 'Nu sunt permise instructiuni de tip LDD dupa ora 12');

else

insert into istoric\_LDD

values (sys.login\_user, sys.database\_name, sys.sysevent, sys.dictionary\_obj\_name, sysdate);

end if;

end;

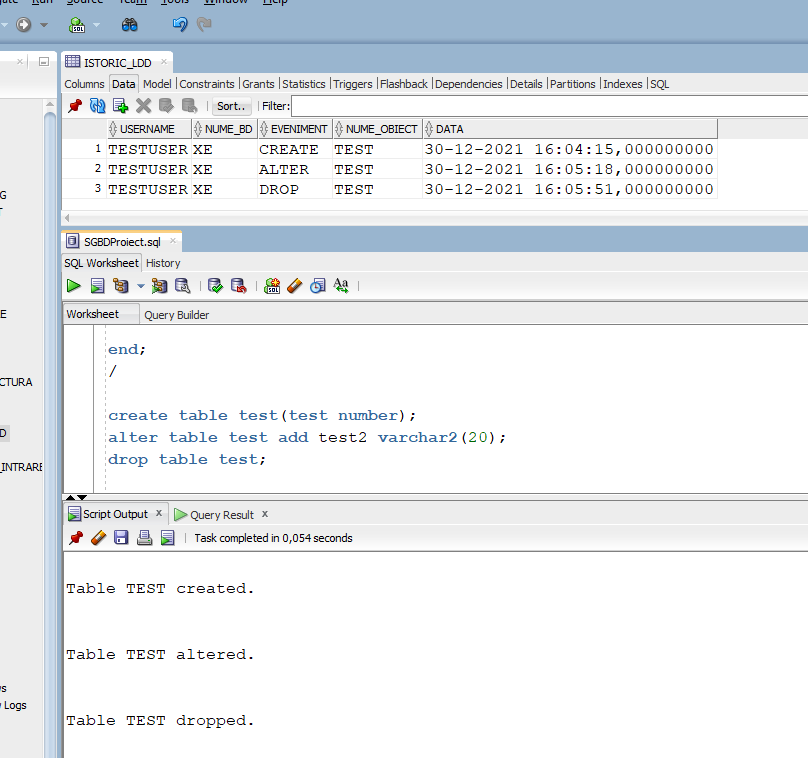
/

După executarea instrucțiunilor în prima jumătate a zilei avem:

create table test(test number);

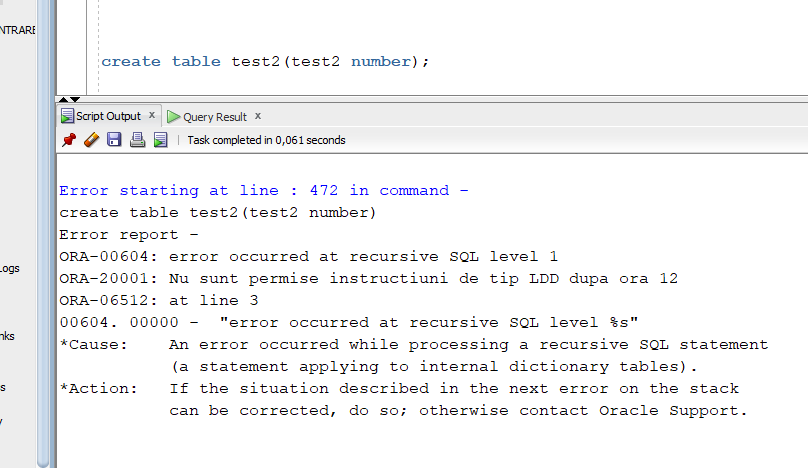
alter table test add test2 varchar2(20);

drop table test;



După executarea instrucțiuni în a doua jumătate a zilei avem:

create table test2(test2 number);



**13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului**

create or replace package ex13 as

procedure p\_ex6;

procedure p\_ex7;

function f\_ex8 (sir varchar2) return number;

procedure p\_ex9 (sir varchar2);

end ex13;

/

create or replace package body ex13 as

procedure p\_ex6 as

type cititori\_type is table of cititori%rowtype;

type biblioteci\_type is table of biblioteci%rowtype index by pls\_integer;

t\_cititori cititori\_type := cititori\_type();

t\_biblioteci biblioteci\_type;

nr number;

rand\_nr pls\_integer;

v\_cititor cititori%rowtype;

v\_biblioteca biblioteci%rowtype;

v\_an number;

v\_exists number;

begin

select c.\*

bulk collect into t\_cititori

from cititori c, (select c.cititor\_id, count(\*) nr\_carti\_citite

from cititori c, fisa\_de\_lectura fl

where c.cititor\_id = fl.cititor\_id

group by c.cititor\_id

having count(\*) >= (select avg(nr\_carti\_citite)

from (select c.cititor\_id, count(\*) nr\_carti\_citite

from cititori c, fisa\_de\_lectura fl

where c.cititor\_id = fl.cititor\_id

group by c.cititor\_id))) t

where c.cititor\_id = t.cititor\_id

order by 1;

dbms\_output.put\_line('Cititori cautati sunt:');

for i in t\_cititori.first..t\_cititori.last loop

v\_cititor := t\_cititori(i);

dbms\_output.put\_line(v\_cititor.nume || ' ' || v\_cititor.prenume || ' ' || v\_cititor.cnp ||

' ' || v\_cititor.telefon || ' ' || v\_cititor.email);

end loop;

dbms\_output.put\_line('');

select b.\*

bulk collect into t\_biblioteci

from biblioteci b, (select biblioteca\_id, count(\*) nr\_cititori

from permis\_de\_intrare

group by biblioteca\_id

having count(\*) = (select min(count(\*)) nr\_cititori

from permis\_de\_intrare

group by biblioteca\_id)) t

where b.biblioteca\_id = t.biblioteca\_id

order by 1;

nr := t\_biblioteci.last;

select dbms\_random.value(1,nr)

into rand\_nr

from dual;

v\_biblioteca := t\_biblioteci(rand\_nr);

dbms\_output.put\_line('Biblioteca aleasa pentru crearea permiselor este '||v\_biblioteca.denumire);

v\_an := extract(year from sysdate);

for i in t\_cititori.first..t\_cititori.last loop

v\_cititor := t\_cititori(i);

select count(\*)

into v\_exists

from permis\_de\_intrare

where cititor\_id = v\_cititor.cititor\_id

and biblioteca\_id = v\_biblioteca.biblioteca\_id;

if v\_exists != 0 then

continue;

end if;

insert into permis\_de\_intrare(permis\_id, anul\_inscrieri, cititor\_id, biblioteca\_id)

values(seq\_permis\_de\_intrare.nextval, v\_an, v\_cititor.cititor\_id, v\_biblioteca.biblioteca\_id);

end loop;

exception

when others then

dbms\_output.put\_line('Ceva nu a mers bine!');

end p\_ex6;

procedure p\_ex7 as

type bibl\_and\_count\_record is record (v\_biblioteca\_id biblioteci.biblioteca\_id%type, v\_count number);

type biblioteci\_and\_count is table of bibl\_and\_count\_record

index by pls\_integer;

cursor c\_angajati (v\_biblioteca\_id angajati.biblioteca\_id%type) is

select \* from angajati where biblioteca\_id = v\_biblioteca\_id;

t\_biblioteci biblioteci\_and\_count;

v\_biblioteca\_id biblioteci.biblioteca\_id%type;

v\_functie\_id functii.functie\_id%type;

max\_sal functii.salariu\_maxim%type;

old\_sal angajati.salariu%type;

new\_sal angajati.salariu%type;

v\_denumire biblioteci.denumire%type;

begin

select biblioteca\_id, count(cititor\_id)

bulk collect into t\_biblioteci

from permis\_de\_intrare

group by biblioteca\_id

having count(cititor\_id) = (select max(count(cititor\_id))

from permis\_de\_intrare

group by biblioteca\_id);

dbms\_output.put\_line('Numarul maxim de cititori inscrisi intr-o biblioteca este '||t\_biblioteci(1).v\_count);

dbms\_output.put\_line('');

dbms\_output.put\_line('Iar aceste biblioteci sunt:');

for i in t\_biblioteci.first..t\_biblioteci.last loop

v\_biblioteca\_id := t\_biblioteci(i).v\_biblioteca\_id;

select denumire into v\_denumire from biblioteci where biblioteca\_id = v\_biblioteca\_id;

dbms\_output.put\_line(v\_denumire);

for v\_ang in c\_angajati(v\_biblioteca\_id) loop

select functie\_id, salariu

into v\_functie\_id, old\_sal

from angajati

where angajat\_id = v\_ang.angajat\_id;

select salariu\_maxim

into max\_sal

from functii

where functie\_id = v\_functie\_id;

new\_sal := old\_sal \* 1.1;

if new\_sal > max\_sal then

new\_sal := max\_sal;

end if;

update angajati

set salariu = new\_sal

where angajat\_id = v\_ang.angajat\_id;

end loop;

end loop;

exception

when others then

dbms\_output.put\_line('Ceva nu a mers bine!');

end p\_ex7;

function f\_ex8 (sir varchar2) return number as

v\_carte\_id carti.carte\_id%type;

v\_res number;

v\_sir varchar2(200);

begin

v\_sir := '%'||lower(sir)||'%';

select carte\_id

into v\_carte\_id

from carti

where lower(titlu) like v\_sir;

for carte in (select ca.titlu, a.nume, a.prenume, sum(co.nr\_exemplare) nr\_exemplare

from carti ca, comenzi co, autori a

where ca.autor\_id = a.autor\_id

and ca.carte\_id = co.carte\_id

and ca.carte\_id = v\_carte\_id

group by ca.titlu, a.nume, a.prenume) loop

v\_res := carte.nr\_exemplare;

dbms\_output.put\_line('Cartea care contine subsirul cautat este: ' || carte.titlu ||' Autorul: ' || carte.nume || ' '

|| carte.prenume);

end loop;

return v\_res;

exception

when no\_data\_found then

dbms\_output.put\_line('Nu exista carti care contin subsirul cautat');

return 0;

when too\_many\_rows then

v\_res := 0;

dbms\_output.put\_line('Cartile care contin subsirul cautat sunt:');

dbms\_output.put\_line('');

for carte in (select ca.titlu, a.nume, a.prenume, sum(co.nr\_exemplare) nr\_exemplare

from carti ca, comenzi co, autori a

where ca.autor\_id = a.autor\_id

and ca.carte\_id = co.carte\_id

and lower(ca.titlu) like v\_sir

group by ca.titlu, a.nume, a.prenume) loop

v\_res := v\_res + carte.nr\_exemplare;

dbms\_output.put\_line(carte.titlu ||' Autorul: ' || carte.nume || ' ' || carte.prenume);

end loop;

return v\_res;

when others then

dbms\_output.put\_line('Ceva nu a mers bine');

return -1;

end f\_ex8;

procedure p\_ex9 (sir varchar2) as

v\_carte\_id carti.carte\_id%type;

v\_sir varchar2(200);

begin

v\_sir := '%'||lower(sir)||'%';

select carte\_id

into v\_carte\_id

from carti

where lower(titlu) like v\_sir;

dbms\_output.put\_line('Istoricul carti cautate este: ');

dbms\_output.put\_line('');

for carte in (select distinct ca.titlu, a.nume nume\_autor, a.prenume prenume\_autor,

b.denumire biblioteca, ci.nume cititor\_nume, ci.prenume cititor\_prenume

from carti ca, fisa\_de\_lectura fl, cititori ci, biblioteci b, autori a

where a.autor\_id = ca.autor\_id

and ca.carte\_id = fl.carte\_id

and fl.cititor\_id = ci.cititor\_id

and fl.biblioteca\_id = b.biblioteca\_id

and ca.carte\_id = v\_carte\_id

order by 1) loop

dbms\_output.put\_line(carte.titlu || ' scrisa de ' || carte.nume\_autor || ' ' || carte.prenume\_autor ||

' citita de ' || carte.cititor\_nume || ' ' || carte.cititor\_prenume || ' la biblioteca ' || carte.biblioteca);

end loop;

exception

when no\_data\_found then

dbms\_output.put\_line('Nu exista carti care contin subsirul cautat');

when too\_many\_rows then

dbms\_output.put\_line('Istoricul cartilor cautate este:');

dbms\_output.put\_line('');

for carte in (select distinct ca.titlu, a.nume nume\_autor, a.prenume prenume\_autor,

b.denumire biblioteca, ci.nume cititor\_nume, ci.prenume cititor\_prenume

from carti ca, fisa\_de\_lectura fl, cititori ci, biblioteci b, autori a

where a.autor\_id = ca.autor\_id

and ca.carte\_id = fl.carte\_id

and fl.cititor\_id = ci.cititor\_id

and fl.biblioteca\_id = b.biblioteca\_id

and lower(ca.titlu) like v\_sir

order by 1) loop

dbms\_output.put\_line(carte.titlu || ' scrisa de ' || carte.nume\_autor || ' ' || carte.prenume\_autor ||

' citita de ' || carte.cititor\_nume || ' ' || carte.cititor\_prenume || ' la biblioteca ' || carte.biblioteca);

end loop;

when others then

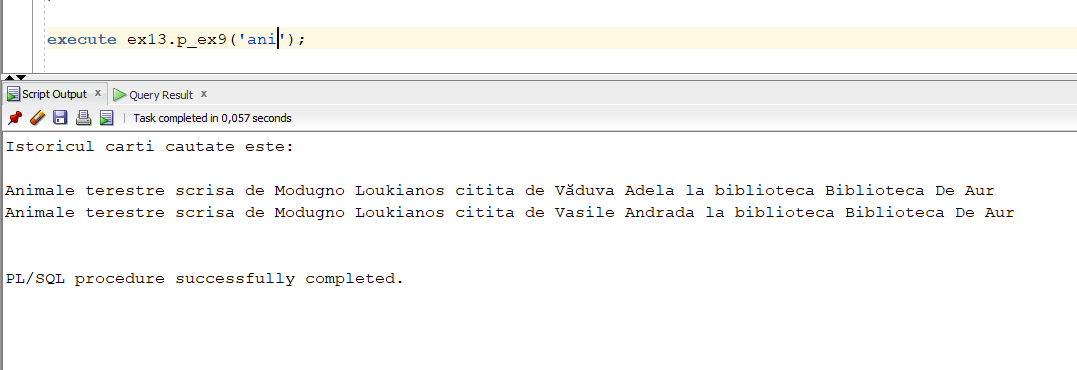
dbms\_output.put\_line('Ceva nu a mers bine');

end p\_ex9;

end ex13;

/

Apelul din pachet al proceduri de la exercițiul 9:



**14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).**

Definiți un pachet care conține:

* Un cursor care va returna subtiupul unei cărți
* O funcție care va returna id-ul unui cititor în funcție de numele și prenumele dat ca parametru
* O funcție care va returna id-ul unei cărți în funcție de titlul dat ca parametru
* O funcție care va returna id-ul unei biblioteci în funcție de denumirea data ca parametru
* O funcție care va determina id-ul unui permis în funcție de id-ul cititorului și id-ul biblioteci dată ca parametru
* O procedură care primește ca parametri numele și prenumele cititorului, titlul unei cărți și denumirea unei biblioteci și va insera în tabelul fisa\_de\_lectura un nou împrumut în funcție de valorile parametrilor care va avea data împrumutului egală cu data curentă iar data de restituire egală cu data de peste 14 zile.
* O procedură care va afișa date despre cărțile care au titlul asemănător cu titlul dat ca parametru.

create or replace package ex14 as

cursor c\_subtip (v\_id carti.carte\_id%type) return subtipuri%rowtype;

function f\_cititor\_id (v\_nume cititori.nume%type, v\_prenume cititori.prenume%type)

return number;

function f\_carte\_id (v\_titlu carti.titlu%type) return number;

function f\_biblioteca\_id(v\_denumire biblioteci.denumire%type) return number;

function f\_permis\_id (v\_cititor\_id cititori.cititor\_id%type, v\_biblioteca\_id biblioteci.biblioteca\_id%type)

return number;

procedure p\_insert\_fisa (v\_nume cititori.nume%type, v\_prenume cititori.prenume%type, v\_titlu carti.titlu%type,

v\_biblioteca biblioteci.denumire%type);

procedure p\_afiseaza\_carti(v\_titlu carti.titlu%type);

end ex14;

/

create or replace package body ex14 as

cursor c\_subtip (v\_id carti.carte\_id%type) return subtipuri%rowtype is

select s.\*

from subtipuri s, catalogare c

where s.subtip\_id = c.subtip\_id

and c.carte\_id = v\_id;

function f\_cititor\_id (v\_nume cititori.nume%type, v\_prenume cititori.prenume%type)

return number as

v\_id cititori.cititor\_id%type;

v\_nume\_aux varchar2(100);

v\_prenume\_aux varchar2(100);

begin

v\_nume\_aux := '%'||lower(v\_nume)||'%';

v\_prenume\_aux := '%'||lower(v\_prenume)||'%';

select cititor\_id

into v\_id

from cititori

where lower(nume) like v\_nume\_aux

and lower(prenume) like v\_prenume\_aux;

return v\_id;

exception

when no\_data\_found then

raise\_application\_error(-20001, 'Nu exista cititori cu numele si prenumele dat!');

when too\_many\_rows then

raise\_application\_error(-20002, 'Exista mai multi cititori cu numele si prenumele dat!');

when others then

raise\_application\_error(-20003, 'Ceva nu a mers bine!');

end f\_cititor\_id;

function f\_carte\_id (v\_titlu carti.titlu%type) return number as

v\_id carti.carte\_id%type;

v\_titlu\_aux varchar2(100);

begin

v\_titlu\_aux := '%'||lower(v\_titlu)||'%';

select carte\_id into v\_id

from carti

where lower(titlu) like v\_titlu\_aux;

return v\_id;

exception

when no\_data\_found then

raise\_application\_error(-20001, 'Nu exista carti cu titlul dat!');

when too\_many\_rows then

raise\_application\_error(-20002, 'Exista mai multe carti cu titlul dat!');

when others then

raise\_application\_error(-20003, 'Ceva nu a mers bine!');

end f\_carte\_id;

function f\_biblioteca\_id(v\_denumire biblioteci.denumire%type) return number as

v\_id biblioteci.biblioteca\_id%type;

v\_denumire\_aux varchar2(100);

begin

v\_denumire\_aux := '%'||lower(v\_denumire)||'%';

select biblioteca\_id into v\_id

from biblioteci

where lower(denumire) like v\_denumire\_aux;

return v\_id;

exception

when no\_data\_found then

raise\_application\_error(-20001, 'Nu exista biblioteci cu denumirea data!');

when too\_many\_rows then

raise\_application\_error(-20002, 'Exista mai multe biblioteci cu denumirea data!');

when others then

raise\_application\_error(-20003, 'Ceva nu a mers bine!');

end f\_biblioteca\_id;

function f\_permis\_id (v\_cititor\_id cititori.cititor\_id%type, v\_biblioteca\_id biblioteci.biblioteca\_id%type)

return number as

v\_id permis\_de\_intrare.permis\_id%type;

begin

select permis\_id

into v\_id

from permis\_de\_intrare

where cititor\_id = v\_cititor\_id

and biblioteca\_id = v\_biblioteca\_id;

return v\_id;

exception

when no\_data\_found then

raise\_application\_error(-20001, 'Nu exista permisul cautat!');

when others then

raise\_application\_error(-20003, 'Ceva nu a mers bine!');

end f\_permis\_id;

procedure p\_insert\_fisa (v\_nume cititori.nume%type, v\_prenume cititori.prenume%type, v\_titlu carti.titlu%type,

v\_biblioteca biblioteci.denumire%type) as

v\_cititor\_id cititori.cititor\_id%type;

v\_carte\_id carti.carte\_id%type;

v\_biblioteca\_id biblioteci.biblioteca\_id%type;

v\_permis\_id permis\_de\_intrare.permis\_id%type;

v\_data\_imprumut date;

v\_data\_restituire date;

begin

v\_cititor\_id := f\_cititor\_id(v\_nume, v\_prenume);

v\_carte\_id := f\_carte\_id(v\_titlu);

v\_biblioteca\_id := f\_biblioteca\_id(v\_biblioteca);

v\_permis\_id := f\_permis\_id(v\_cititor\_id, v\_biblioteca\_id);

v\_data\_imprumut := sysdate;

v\_data\_restituire := v\_data\_imprumut + interval '14' day;

insert into fisa\_de\_lectura(fisa\_id, permis\_id, cititor\_id, biblioteca\_id, carte\_id,

data\_imprumut, data\_restituire)

values(seq\_fisa\_de\_lectura.nextval, v\_permis\_id, v\_cititor\_id, v\_biblioteca\_id, v\_carte\_id,

v\_data\_imprumut, v\_data\_restituire);

end p\_insert\_fisa;

procedure p\_afiseaza\_carti(v\_titlu carti.titlu%type) as

type carti\_type is table of carti%rowtype index by pls\_integer;

type carte\_autor\_editura is record(editura edituri.denumire%type, nume\_autor autori.nume%type,

prenume\_autor autori.prenume%type);

t\_carti carti\_type;

v\_titlu\_aux varchar2(100);

v\_carte carti%rowtype;

v\_cae carte\_autor\_editura;

begin

v\_titlu\_aux := '%' || lower(v\_titlu) || '%';

select \*

bulk collect into t\_carti

from carti

where lower(titlu) like v\_titlu\_aux;

if t\_carti.count = 0 then

raise no\_data\_found;

end if;

for i in t\_carti.first..t\_carti.last loop

v\_carte := t\_carti(i);

select e.denumire, a.nume, a.prenume

into v\_cae

from carti c, autori a, edituri e

where c.autor\_id = a.autor\_id

and c.editura\_id = e.editura\_id

and c.carte\_id = v\_carte.carte\_id;

dbms\_output.put\_line('Titlul: ' || v\_carte.titlu);

dbms\_output.put\_line('Editura: ' || v\_cae.editura);

dbms\_output.put\_line('An aparitie: '||v\_carte.an\_aparitie);

dbms\_output.put\_line('Autorul: '||v\_cae.nume\_autor||' '||v\_cae.prenume\_autor);

dbms\_output.put\_line('Tipul: '||v\_carte.tip\_carte);

dbms\_output.put('Subtipurile: ');

for subtip in c\_subtip(v\_carte.carte\_id) loop

dbms\_output.put(subtip.denumire||' ');

end loop;

dbms\_output.new\_line;

dbms\_output.put\_line('');

end loop;

exception

when no\_data\_found then

dbms\_output.put\_line('Nu am gasit carti');

when others then

dbms\_output.put\_line('Ceva nu a mers bine');

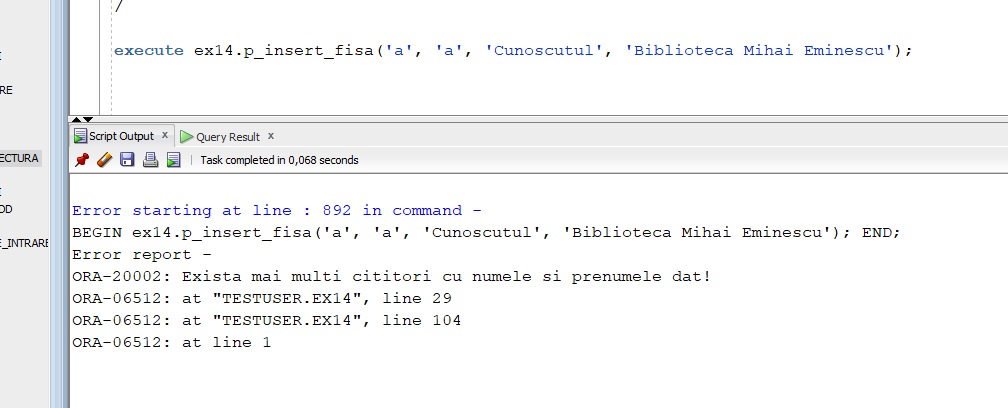
end p\_afiseaza\_carti;

end ex14;

/

**Pentru procedura ex14.p\_insert\_fisa avem cazurile:**

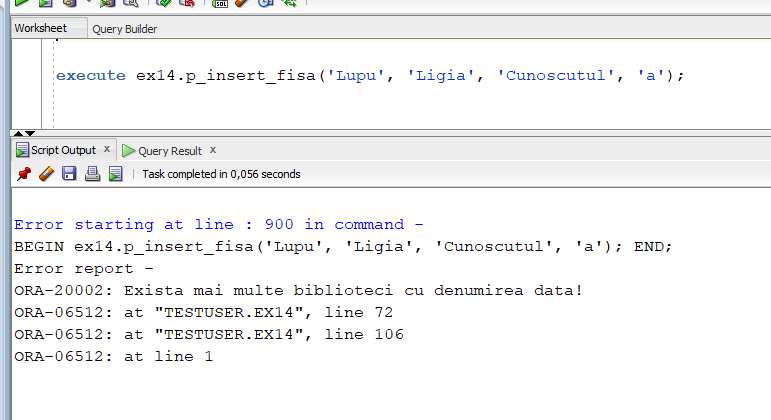
Cazul când nu am putut găsi un cititor:



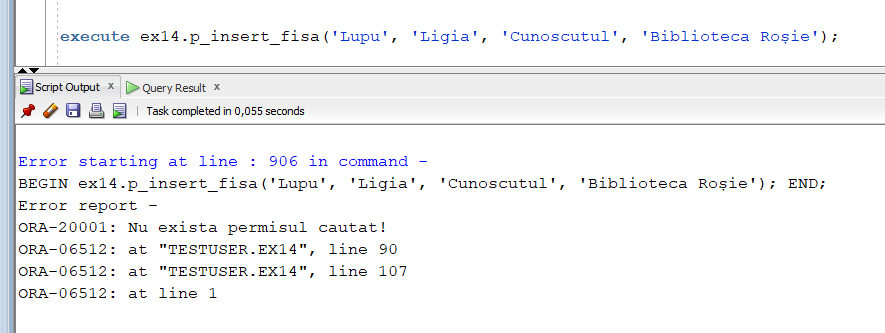
Cazul când nu am putut găsi o carte:



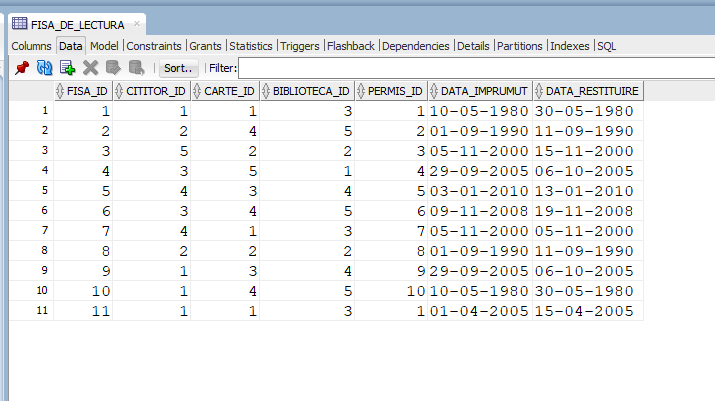
Cazul când nu am putut găsi o bibliotecă:

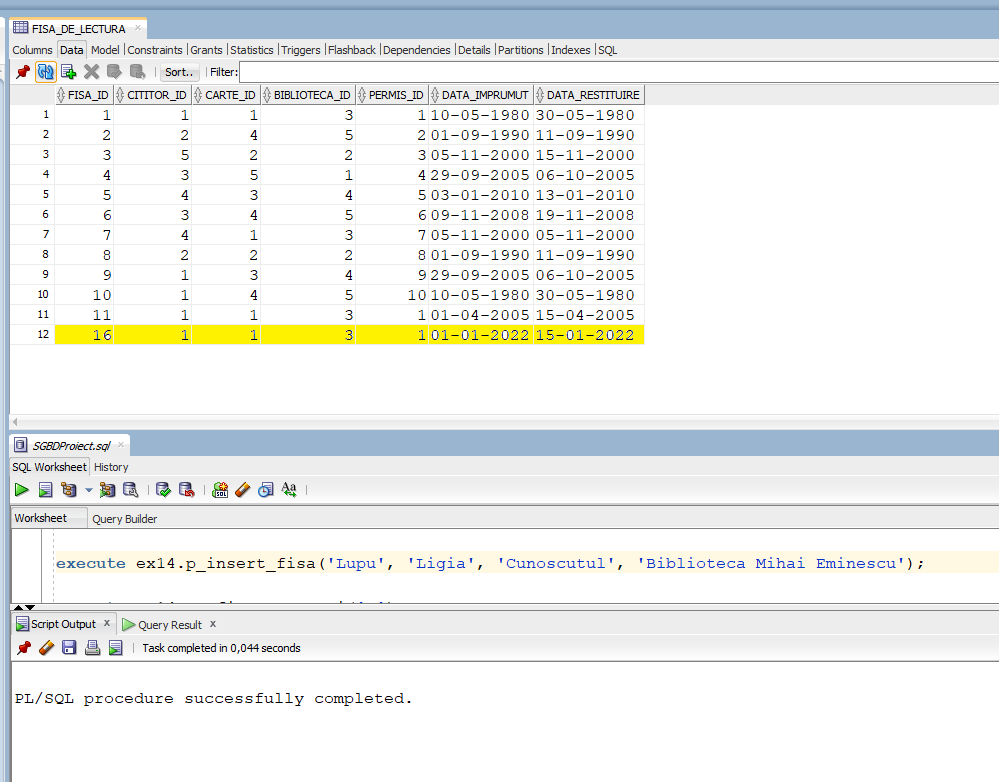


Cazul când nu am putut găsi un permis:



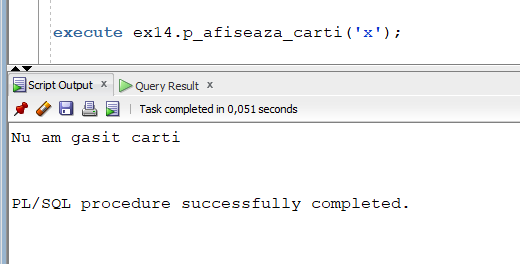
Cazul când am putut insera date:





**Pentru procedura ex14.** **p\_afiseaza\_carti avem cazurile:**

Când nu găsim cărți în baza de date:



Când găsim cărți în baza de date:

