Centrul de adopție

pentru animalele de companie "Happy Puppy"

Tache Daria Elena

Grupa 234

|  |  |  |
| --- | --- | --- |
|  |  |  |

*Cuprins*

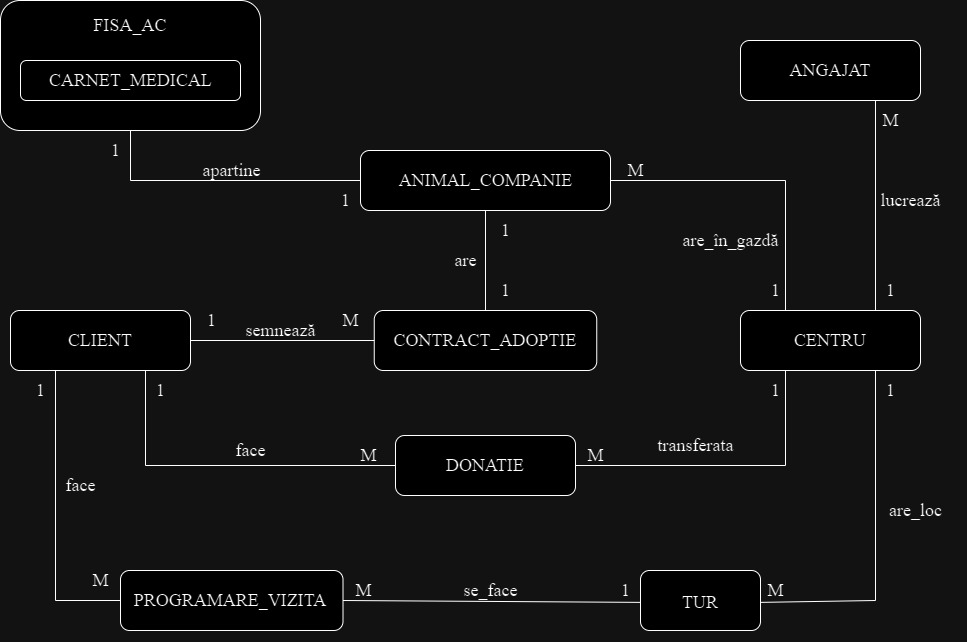
1. [Prezentarea pe scurt a bazei de date.](#Marcaj în document1)
2. [Diagrama entitate-relație (ERD).](#Marcaj în document2)
3. [Diagrama conceptuală a modelului propus, având integrate toate atributele necesare: entitățile, relațiile și atributele definite în limba română.](#Marcaj în document3)
4. [Implementarea în Oracle a diagramei conceptuale: definirea tabelelor și a constrângerilor de integritate necesare.](#Marcaj în document4)
5. [Adăugarea informațiilor coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).](#Marcaj în document5)
6. [Formularea enunțului unei probleme rezolvate folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate.](#Marcaj în document6)
7. [Formularea enunțului unei probleme rezolvate folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor.](#Marcaj în document7)
8. [Formularea enunțului unei probleme rezolvate folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definirea a minim 2 excepții proprii. Apelul subprogramului astfel încât au fost evidențiate toate cazurile definite și tratate.](#Marcaj în document8)
9. [Formularea enunțului unei probleme rezolvate folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratarea tuturor excepțiilor care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelul subprogramului astfel încât au fot evidențiate toate cazurile tratate.](#Marcaj în document9)
10. [Definirea și declanșarea unui trigger de tip LMD la nivel de comandă.](#Marcaj în document10)
11. [Definirea și declanșarea unui trigger de tip LMD la nivel de linie.](#Marcaj în document11)
12. [Definirea și declanșarea unui trigger de tip LDD.](#Marcaj în document12)
13. *Prezentarea pe scurt a bazei de date:*

Proiectul are ca scop crearea unei baze de date pentru fundația de adopție al animăluțelor de companie “Happy Puppy”, cu cabinete în întreaga țară. În acest fel, suntem mai aproape de persoanele ce vor să ofere micilor prieteni șansa la o viață alături de o familie iubitoare.

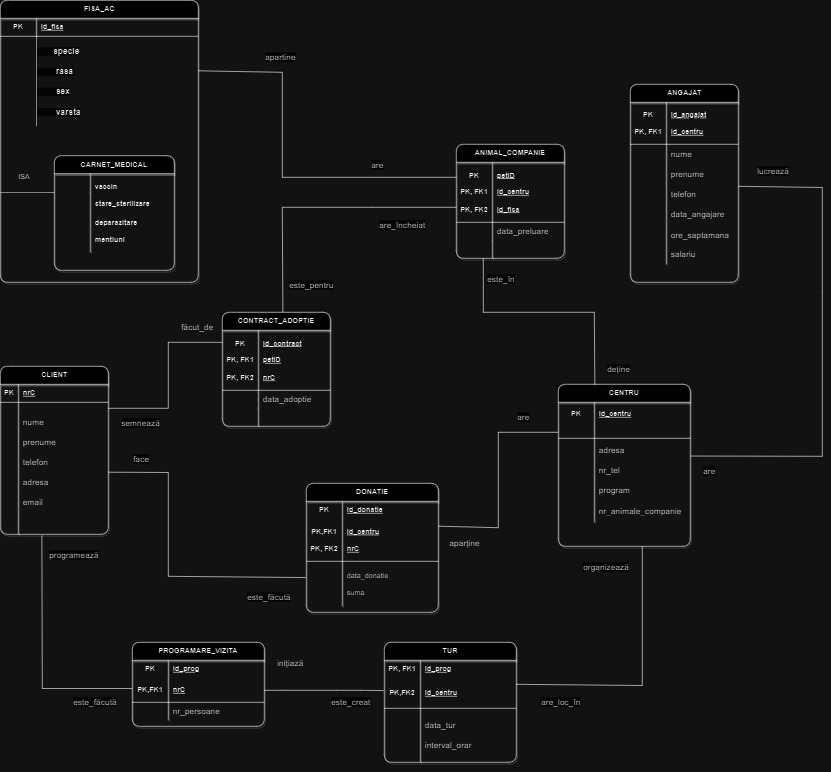
Centrele noastre vă oferă mai multe opțiuni: puteți adopta un animăluț, puteți dona pentru micii amici pufoși sau puteți veni cu copiii în baza unei programări la joacă!

Procesul de adopție este mai ușor ca niciodată prin contractul de adopție, iar colegii noștri vă vor fi alături la fiecare pas. La final veți primi carnetul medical al noului vostru animăluț de companie, cu vaccinurile la zi!

1. *Diagrama entitate-**relație (ERD):*



1. *Diagrama* *conceptuală a modelului propus, având integrate toate atributele necesare: entitățile, relațiile și atributele definite în limba română.*



1. *Implementarea în Oracle a diagramei conceptuale: definirea tabelelelor și a* *constrângerilor de integritate necesare.*

CREATE TABLE FISA\_AC (

id\_fisa NUMBER PRIMARY KEY,

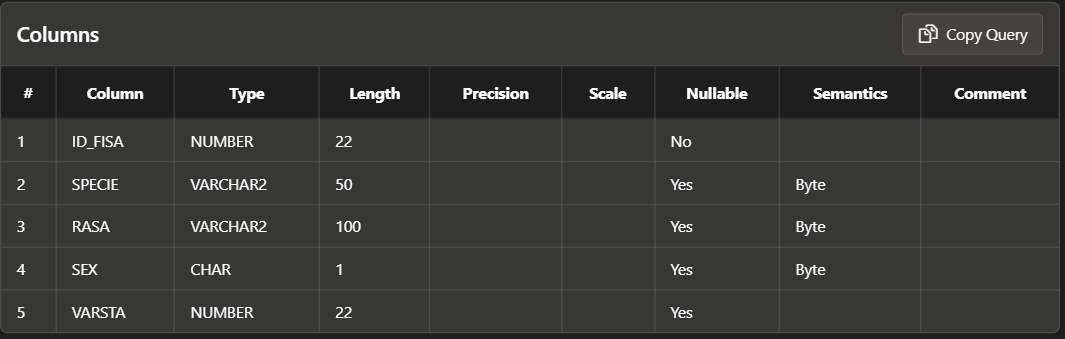
specie VARCHAR2(50),

rasa VARCHAR2(100),

sex CHAR(1),

varsta NUMBER

);



CREATE TABLE CARNET\_MEDICAL(

id\_fisa NUMBER,

vaccin VARCHAR2(500),

stare\_sterilizare CHAR(3) CHECK (UPPER(stare\_sterilizare) IN ('DA', 'NU')),

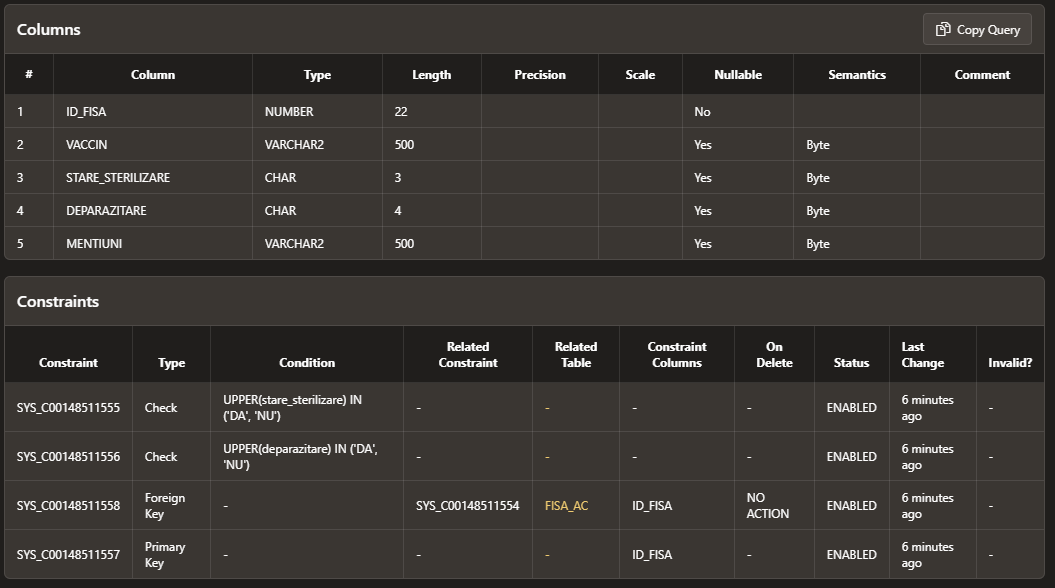
deparazitare CHAR(4) CHECK (UPPER(deparazitare) IN ('DA', 'NU')),

mentiuni VARCHAR2(500),

PRIMARY KEY(id\_fisa),

FOREIGN KEY(id\_fisa) REFERENCES FISA\_AC(id\_fisa)

);



CREATE TABLE CENTRU (

id\_centru NUMBER PRIMARY KEY,

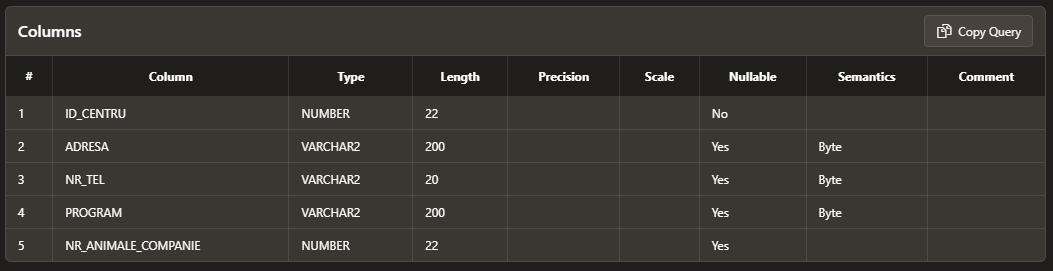
adresa VARCHAR2(200),

nr\_tel VARCHAR2(20),

program VARCHAR2(200),

nr\_animale\_companie NUMBER

);



CREATE TABLE ANIMAL\_COMPANIE (

petID NUMBER PRIMARY KEY,

id\_centru NUMBER,

id\_fisa NUMBER,

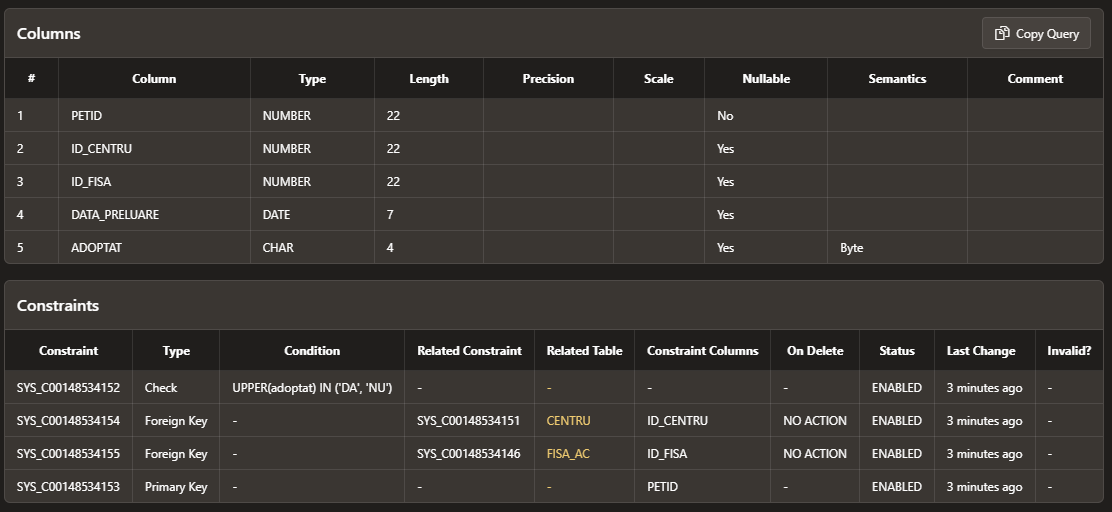
data\_preluare DATE,

adoptat CHAR(4) CHECK (UPPER(adoptat) IN ('DA', 'NU')),

FOREIGN KEY (id\_centru) REFERENCES CENTRU(id\_centru),

FOREIGN KEY (id\_fisa) REFERENCES FISA\_AC(id\_fisa)

);



CREATE TABLE CLIENT(

nrC NUMBER PRIMARY KEY,

nume VARCHAR2(20),

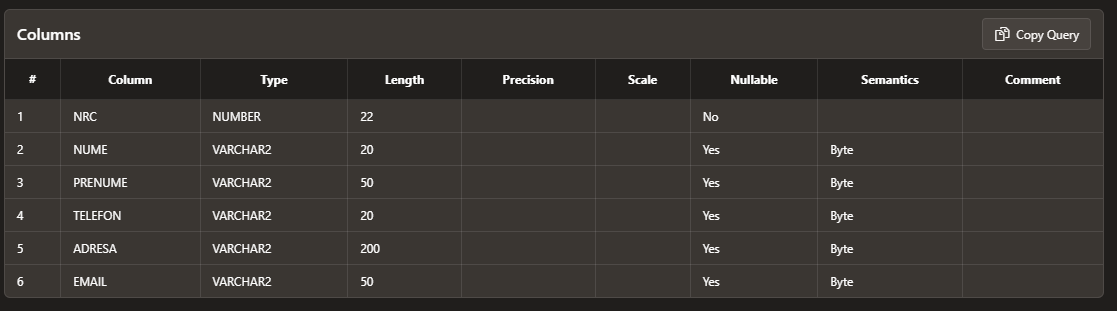
prenume VARCHAR2(50),

telefon VARCHAR2(20),

adresa VARCHAR2(200),

email VARCHAR2(50)

);



CREATE TABLE PROGRAMARE\_VIZITA(

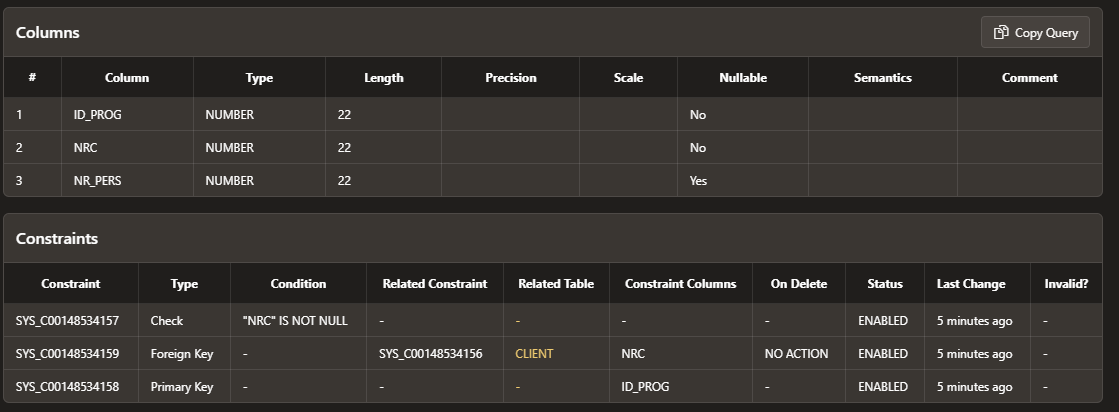
id\_prog NUMBER PRIMARY KEY,

nrC NUMBER NOT NULL,

nr\_pers NUMBER,

FOREIGN KEY (nrC) REFERENCES CLIENT (nrC)

);



CREATE TABLE TUR(

id\_prog NUMBER,

id\_centru NUMBER,

data\_tur DATE,

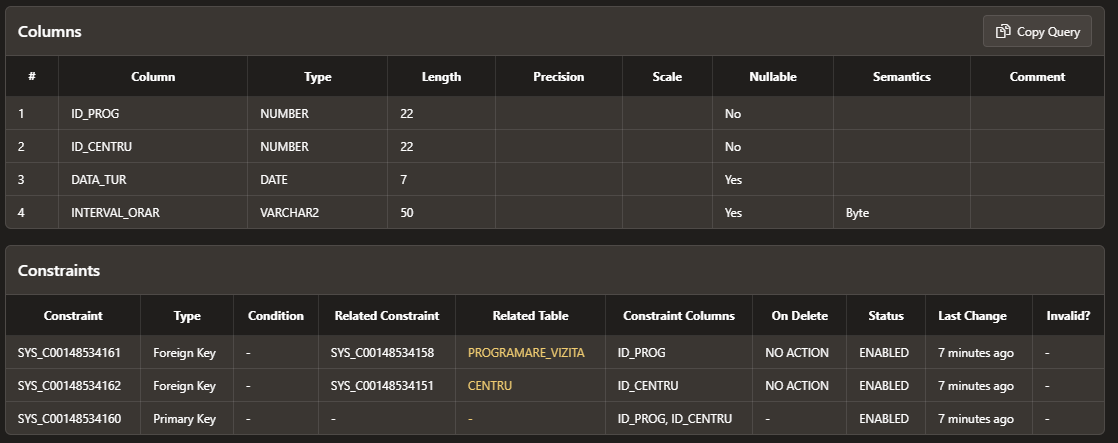
interval\_orar VARCHAR2(50),

PRIMARY KEY(id\_prog, id\_centru),

FOREIGN KEY(id\_prog) REFERENCES PROGRAMARE\_VIZITA(id\_prog),

FOREIGN KEY (id\_centru) REFERENCES CENTRU(id\_centru)

);



CREATE TABLE DONATIE(

id\_donatie NUMBER PRIMARY KEY,

id\_centru NUMBER,

nrC NUMBER,

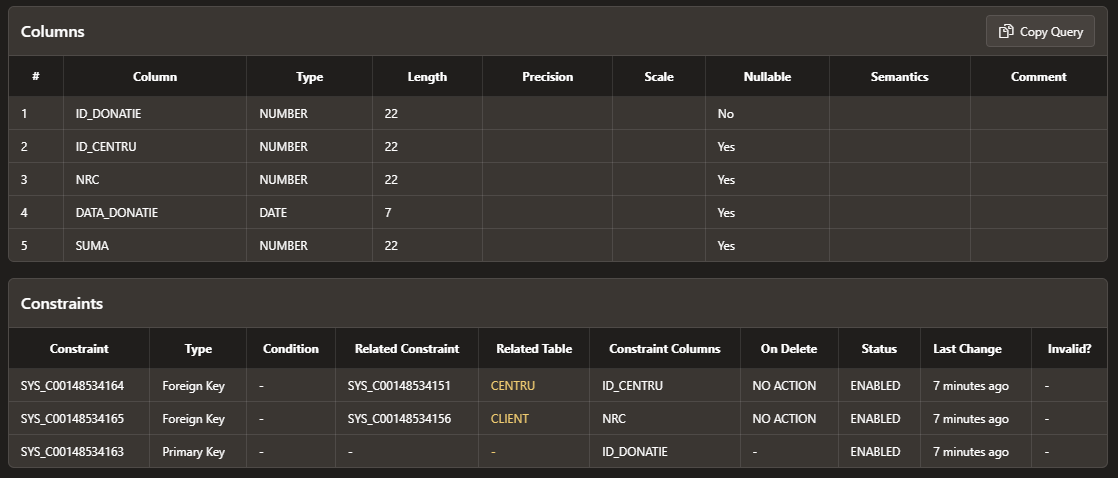
data\_donatie DATE,

suma NUMBER,

FOREIGN KEY(id\_centru) REFERENCES CENTRU(id\_centru),

FOREIGN KEY(nrC) REFERENCES CLIENT(nrC)

);



CREATE TABLE CONTRACT\_ADOPTIE(

id\_contract NUMBER PRIMARY KEY,

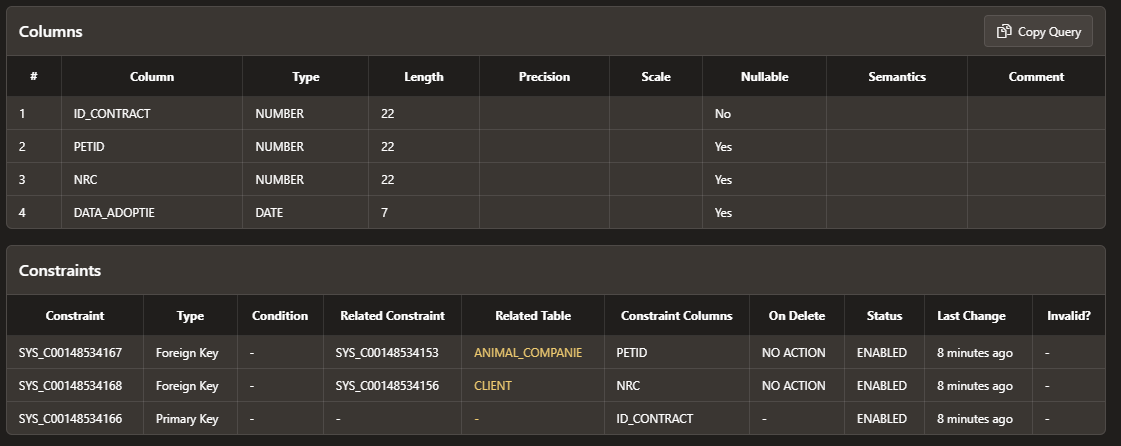
petID NUMBER,

nrC NUMBER,

data\_adoptie DATE,

FOREIGN KEY (petID) REFERENCES ANIMAL\_COMPANIE(petID),

FOREIGN KEY (nrC) REFERENCES CLIENT(nrC)

); 

CREATE TABLE ANGAJAT(

id\_angajat NUMBER PRIMARY KEY,

id\_centru NUMBER,

nume VARCHAR2(20),

prenume VARCHAR2(50),

telefon VARCHAR2(20),

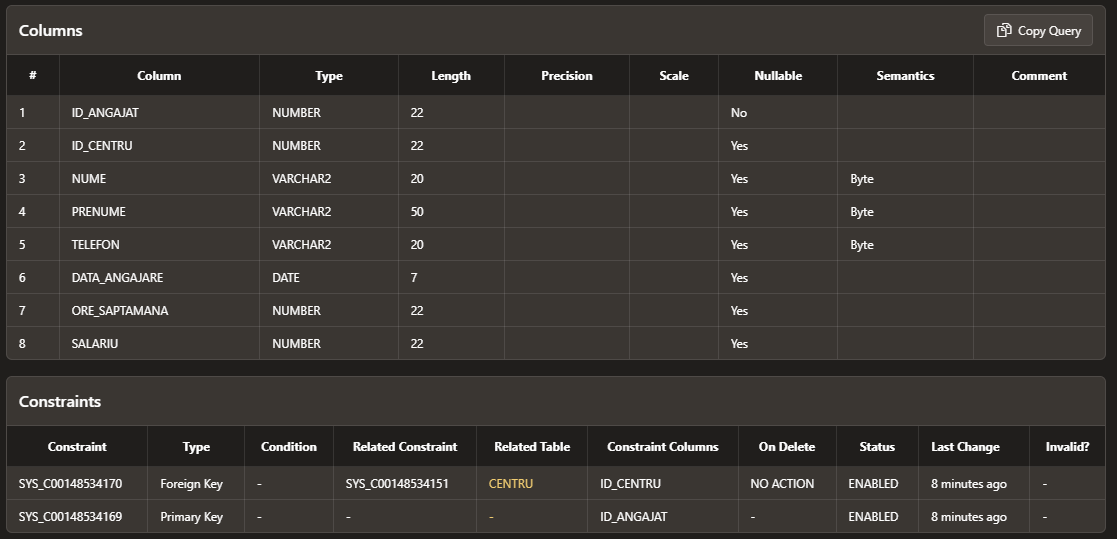
data\_angajare DATE,

ore\_saptamana NUMBER,

salariu NUMBER,

FOREIGN KEY (id\_centru) REFERENCES CENTRU(id\_centru)

);



1. *Adăugarea informațiilor coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).*

CREATE SEQUENCE seq\_centreID

INCREMENT BY 1

START WITH 100

MINVALUE 1

MAXVALUE 5000

NOCYCLE;

CREATE SEQUENCE seq\_petID

INCREMENT BY 1

START WITH 1400

MINVALUE 1

MAXVALUE 250000

NOCYCLE;

CREATE SEQUENCE seq\_id\_fisa

START WITH 1

INCREMENT BY 1

NOCYCLE

CACHE 40;

INSERT INTO CENTRU

(id\_centru, adresa, nr\_tel, program, nr\_animale\_companie)

VALUES

(seq\_centreID.NEXTVAL, 'Str Alexandru cel Mare, 21', '0726248310', '08:00 - 18:00', 0 );

INSERT INTO CENTRU

(id\_centru, adresa, nr\_tel, program, nr\_animale\_companie)

VALUES

(seq\_centreID.NEXTVAL, 'Strada Victoriei, 10', '0723123456', '09:00 - 19:00', 0);

INSERT INTO CENTRU

(id\_centru, adresa, nr\_tel, program, nr\_animale\_companie)

VALUES

(seq\_centreID.NEXTVAL, 'Bulevardul Independenței, 5', '0732112233', '10:00 - 20:00', 0);

INSERT INTO CENTRU

(id\_centru, adresa, nr\_tel, program, nr\_animale\_companie)

VALUES

(seq\_centreID.NEXTVAL, 'Strada Aviatorilor, 15', '0712345678', '08:30 - 18:30', 0);

INSERT INTO CENTRU

(id\_centru, adresa, nr\_tel, program, nr\_animale\_companie)

VALUES

(seq\_centreID.NEXTVAL, 'Bulevardul Dacia, 8', '0755555555', '10:30 - 20:30', 0);

DECLARE

TYPE SpeciiType IS TABLE OF VARCHAR2(20);

TYPE VarstaType IS TABLE OF NUMBER;

TYPE RaseType IS TABLE OF VARCHAR2(50);

TYPE CheckType IS TABLE OF VARCHAR2(3);

TYPE MentiuniType IS TABLE OF VARCHAR2(20);

TYPE GenderType IS TABLE OF VARCHAR2(3);

TYPE VaccinType IS TABLE OF VARCHAR(20);

TYPE DatePreluareType IS TABLE OF DATE INDEX BY PLS\_INTEGER;

v\_specii SpeciiType := SpeciiType('Caine', 'Pisica', 'Iepure', 'Hamster', 'Papagal');

v\_varsta VarstaType := VarstaType(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12);

v\_rase\_caine RaseType := RaseType('Labrador', 'Ciobanesc German', 'Bullog');

v\_rase\_pisica RaseType := RaseType('Siameza', 'Albastru de Rusia', 'British Shorthair');

v\_rase\_iepure RaseType := RaseType('Pitic', 'Belier');

v\_rase\_hamster RaseType := RaseType('Pitic Roborovsky');

v\_check CheckType := CheckType('DA', 'NU');

v\_mentiuni MentiuniType := MentiuniType('Nicio mentiune', 'Are alergii', 'Iubitor');

v\_gender GenderType := GenderType('F', 'M');

v\_vaccin VaccinType := VaccinType('Vaccin A', 'Vaccin B', 'Vaccin C', 'Vaccin D');

v\_date\_preluare DatePreluareType;

v\_nr\_specii CONSTANT NUMBER := v\_specii.COUNT;

v\_nr\_rase\_caine CONSTANT NUMBER := v\_rase\_caine.COUNT;

v\_nr\_rase\_pisica CONSTANT NUMBER := v\_rase\_pisica.COUNT;

v\_nr\_rase\_iepure CONSTANT NUMBER := v\_rase\_iepure.COUNT;

v\_nr\_rase\_hamster CONSTANT NUMBER := v\_rase\_hamster.COUNT;

v\_nr\_check CONSTANT NUMBER := v\_check.COUNT;

v\_nr\_mentiuni CONSTANT NUMBER := v\_mentiuni.COUNT;

v\_nr\_gender CONSTANT NUMBER := v\_gender.COUNT;

v\_nr\_vaccin CONSTANT NUMBER := v\_vaccin.COUNT;

v\_random\_specie VARCHAR2(20);

v\_random\_varsta NUMBER;

v\_random\_rasa VARCHAR2(50);

v\_random\_check1 VARCHAR2(3);

v\_random\_check2 VARCHAR2(3);

v\_random\_mentiune VARCHAR2(20);

v\_random\_gender VARCHAR(3);

v\_random\_vaccin VARCHAR(20);

v\_random\_data DATE;

v\_counter NUMBER := 0;

v\_id\_fisa NUMBER;

v\_id\_centru NUMBER;

v\_petID NUMBER;

v\_nr\_centre NUMBER;

v\_centru\_curent NUMBER;

BEGIN

-- Introduc date in tabel:

v\_date\_preluare(1) := TO\_DATE('2023-01-01', 'YYYY-MM-DD');

v\_date\_preluare(2) := TO\_DATE('2023-02-15', 'YYYY-MM-DD');

v\_date\_preluare(3) := TO\_DATE('2023-03-10', 'YYYY-MM-DD');

v\_date\_preluare(4) := TO\_DATE('2023-04-22', 'YYYY-MM-DD');

v\_date\_preluare(5) := TO\_DATE('2023-05-05', 'YYYY-MM-DD');

v\_date\_preluare(6) := TO\_DATE('2023-06-18', 'YYYY-MM-DD');

v\_date\_preluare(7) := TO\_DATE('2023-07-03', 'YYYY-MM-DD');

v\_date\_preluare(8) := TO\_DATE('2023-08-14', 'YYYY-MM-DD');

v\_date\_preluare(9) := TO\_DATE('2023-09-27', 'YYYY-MM-DD');

v\_date\_preluare(10) := TO\_DATE('2023-10-09', 'YYYY-MM-DD');

SELECT COUNT(\*) INTO v\_nr\_centre FROM CENTRU;

WHILE v\_counter < 16 LOOP

-- Se obtine id-ul unui centru random

SELECT id\_centru INTO v\_centru\_curent FROM CENTRU WHERE ROWNUM = 1 ORDER BY DBMS\_RANDOM.VALUE;

-- Se obține id-ul unei fise disponibile

v\_id\_fisa := seq\_id\_fisa.NEXTVAL;

-- Se obține un nou id pentru animal

v\_petID := seq\_petID.NEXTVAL;

v\_random\_specie := v\_specii(DBMS\_RANDOM.VALUE(1, v\_nr\_specii));

-- Alegere aleatoare pentru rasa în funcție de specie

CASE v\_random\_specie

WHEN 'Caine' THEN v\_random\_rasa := v\_rase\_caine(DBMS\_RANDOM.VALUE(1, v\_nr\_rase\_caine));

WHEN 'Pisica' THEN v\_random\_rasa := v\_rase\_pisica(DBMS\_RANDOM.VALUE(1, v\_nr\_rase\_pisica));

WHEN 'Iepure' THEN v\_random\_rasa := v\_rase\_iepure(DBMS\_RANDOM.VALUE(1, v\_nr\_rase\_iepure));

WHEN 'Hamster' THEN v\_random\_rasa := v\_rase\_hamster(DBMS\_RANDOM.VALUE(1, v\_nr\_rase\_hamster));

ELSE v\_random\_rasa := NULL;

END CASE;

-- Alegere aleatoare pentru Check, Varsta, Gender, Vaccin, Mentiuni si Data

v\_random\_check1 := v\_check(DBMS\_RANDOM.VALUE(1, v\_nr\_check));

v\_random\_check2 := v\_check(DBMS\_RANDOM.VALUE(1, v\_nr\_check));

v\_random\_varsta := v\_varsta(DBMS\_RANDOM.VALUE(1,12));

v\_random\_gender := v\_gender(DBMS\_RANDOM.VALUE(1, v\_nr\_gender));

v\_random\_vaccin := v\_vaccin(DBMS\_RANDOM.VALUE(1, v\_nr\_vaccin));

v\_random\_mentiune := v\_mentiuni(DBMS\_RANDOM.VALUE(1, v\_nr\_mentiuni));

v\_random\_data := v\_date\_preluare(DBMS\_RANDOM.VALUE(1,10));

INSERT INTO FISA\_AC

(id\_fisa, specie, rasa, sex, varsta)

VALUES

(v\_id\_fisa, v\_random\_specie, v\_random\_rasa, v\_random\_gender, v\_random\_varsta);

INSERT INTO CARNET\_MEDICAL

(id\_fisa, vaccin, stare\_sterilizare, deparazitare, mentiuni)

VALUES

(v\_id\_fisa, v\_random\_vaccin, v\_random\_check1, v\_random\_check2, v\_random\_mentiune);

INSERT INTO ANIMAL\_COMPANIE

(petID, id\_centru, id\_fisa, data\_preluare, adoptat)

VALUES

(v\_petID, v\_centru\_curent, v\_id\_fisa, v\_random\_data, 'NU');

-- După inserarea în ANIMAL\_COMPANIE

UPDATE CENTRU

SET nr\_animale\_companie = nr\_animale\_companie + 1

WHERE id\_centru = v\_centru\_curent;

v\_counter := v\_counter + 1;

END LOOP;

END;

/

1. *Formularea* *enunțului unei probleme rezolvate folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate.*

/\*

Clientii "Star" sunt clientii care au donat, au planificat o vizita si au adoptat.

Retinand in 3 tipuri diferite de colectii de date id-urile clientilor din tabela de donatie,

planificare\_vizita si contract\_adoptie, voi afisa pe ecran cati clienti "Star" avem

Raspuns: 1

\*/

CREATE OR REPLACE PROCEDURE Exercitiul6 IS

TYPE DonatieType IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

TYPE VizitaType IS TABLE OF NUMBER;

TYPE AdoptieType IS VARRAY(50) OF NUMBER;

TYPE DetaliiType IS RECORD (

nume CLIENT.nume%TYPE,

prenume CLIENT.prenume%TYPE

);

v\_detalii DetaliiType;

v\_donatii DonatieType;

v\_vizite VizitaType := VizitaType();

v\_adoptii AdoptieType := AdoptieType();

v\_nrC NUMBER;

v\_star\_clients NUMBER := 0;

BEGIN

SELECT DISTINCT nrC

BULK COLLECT INTO v\_donatii

FROM DONATIE;

FOR i IN (SELECT DISTINCT nrC FROM PROGRAMARE\_VIZITA) LOOP

v\_vizite.EXTEND;

v\_vizite(v\_vizite.LAST) := i.nrC;

END LOOP;

FOR i IN (SELECT DISTINCT nrC FROM CONTRACT\_ADOPTIE) LOOP

v\_adoptii.EXTEND;

v\_adoptii(v\_adoptii.LAST) := i.nrC;

END LOOP;

FOR i IN (SELECT DISTINCT nrC FROM CLIENT) LOOP

v\_nrC := i.nrC;

-- Vedem dacă nrC există în toate cele trei colecții

IF v\_donatii.EXISTS(v\_nrC) AND v\_vizite.EXISTS(v\_nrC) AND v\_adoptii.EXISTS(v\_nrC) THEN

SELECT nume, prenume INTO v\_detalii.nume, v\_detalii.prenume

FROM CLIENT

WHERE nrC = v\_nrC;

DBMS\_OUTPUT.PUT\_LINE('Client ' || v\_nrC || ', ' || v\_detalii.nume || ' ' || v\_detalii.prenume ||' este un Star Client!');

v\_star\_clients := v\_star\_clients + 1;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Numar total de Star Clients: ' || v\_star\_clients);

END Exercitiul6;

/

-- Apelul subprogramului:

BEGIN

Exercitiul6;

END;

/

1. *Formularea enunțului unei probleme rezolvate folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor.*

CREATE OR REPLACE PROCEDURE CalculSumaDonatii IS

-- Cursorul implicit pentru Centre

CURSOR centru\_cursor IS

SELECT id\_centru, adresa FROM CENTRU;

-- Cursorul pentru Donații

CURSOR donatie\_cursor (p\_id\_centru\_donatie NUMBER) IS

SELECT suma FROM DONATIE

WHERE id\_centru = p\_id\_centru\_donatie;

v\_id\_centru CENTRU.id\_centru%TYPE;

v\_adresa\_centru CENTRU.adresa%TYPE;

v\_suma\_totala NUMBER := 0;

v\_suma\_donatie NUMBER;

BEGIN

OPEN centru\_cursor;

LOOP

FETCH centru\_cursor INTO v\_id\_centru, v\_adresa\_centru;

EXIT WHEN centru\_cursor%NOTFOUND;

OPEN donatie\_cursor(v\_id\_centru);

LOOP

FETCH donatie\_cursor INTO v\_suma\_donatie;

EXIT WHEN donatie\_cursor%NOTFOUND;

v\_suma\_totala := v\_suma\_totala + v\_suma\_donatie;

END LOOP;

CLOSE donatie\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Centrul cu adresa: ' || v\_adresa\_centru);

DBMS\_OUTPUT.PUT\_LINE('Suma totală donată: ' || v\_suma\_totala);

v\_suma\_totala := 0;

END LOOP;

CLOSE centru\_cursor;

END CalculSumaDonatii;

/

select \* from donatie;

BEGIN

CalculSumaDonatii();

END;

/

1. *Formularea enunțului unei probleme rezolvate folosind un subprogram* *stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definirea a minim 2 excepții proprii. Apelul subprogramului astfel încât au fost evidențiate toate cazurile definite și tratate.*

* Am facut o functie care programeaza tururi.

CREATE OR REPLACE PROCEDURE ProgramareTur(

p\_nume CLIENT.nume%TYPE,

p\_prenume CLIENT.prenume%TYPE,

p\_telefon CLIENT.telefon%TYPE,

p\_id\_centru NUMBER,

p\_nr\_persoane PROGRAMARE\_VIZITA.nr\_pers%TYPE,

p\_ora\_inceput NUMBER,

p\_ora\_final NUMBER

) IS

v\_nrC CLIENT.nrC%TYPE;

v\_id\_prog PROGRAMARE\_VIZITA.id\_prog%TYPE;

v\_data\_tur TUR.data\_tur%TYPE;

v\_durata\_tur NUMBER;

v\_aux VARCHAR(20);

v\_verif NUMBER;

FORMAT\_INVALID EXCEPTION;

PRAGMA EXCEPTION\_INIT(FORMAT\_INVALID, -2292);

DURATA\_DEPASITA EXCEPTION;

PRAGMA EXCEPTION\_INIT(DURATA\_DEPASITA, -2293);

BEGIN

-- Verificam daca centrul exista

SELECT 1

INTO v\_verif

FROM CENTRU

WHERE id\_centru = p\_id\_centru;

BEGIN

-- Se verifică dacă clientul există deja în tabelul CLIENT

SELECT nrC INTO v\_nrC

FROM CLIENT

WHERE nume = p\_nume AND prenume = p\_prenume AND telefon = p\_telefon;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_nrC := seq\_nrC.NEXTVAL;

INSERT INTO CLIENT(nrC, nume, prenume, telefon)

VALUES (v\_nrC, p\_nume, p\_prenume, p\_telefon);

END;

IF p\_ora\_inceput > p\_ora\_final THEN

RAISE FORMAT\_INVALID;

END IF;

v\_durata\_tur := p\_ora\_final - p\_ora\_inceput;

IF v\_durata\_tur > 3 THEN

RAISE DURATA\_DEPASITA;

END IF;

v\_id\_prog := seq\_id\_prog.NEXTVAL;

INSERT INTO PROGRAMARE\_VIZITA(id\_prog, nrC, nr\_pers)

VALUES (v\_id\_prog, v\_nrC, p\_nr\_persoane);

v\_aux := TO\_CHAR(p\_ora\_inceput) || '-' || TO\_CHAR(p\_ora\_final);

INSERT INTO TUR(id\_prog, id\_centru, data\_tur, interval\_orar)

VALUES (v\_id\_prog, p\_id\_centru, SYSDATE, v\_aux);

DBMS\_OUTPUT.PUT\_LINE('Programare pentru tur realizată cu succes!');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Id\_centru NU există în tabela CENTRU.');

WHEN FORMAT\_INVALID THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || 'Ora de final nu poate fi precedentă orei de început.');

WHEN DURATA\_DEPASITA THEN

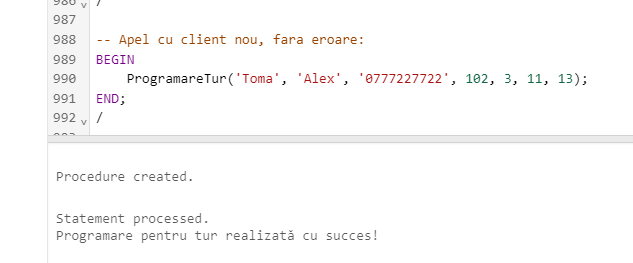
DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || 'Durata turului nu poate depăși 3 ore.');

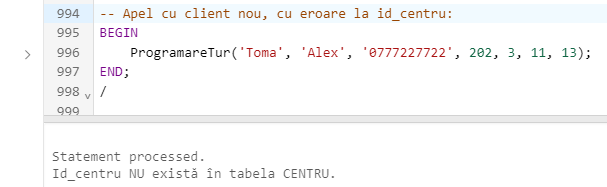
WHEN OTHERS THEN

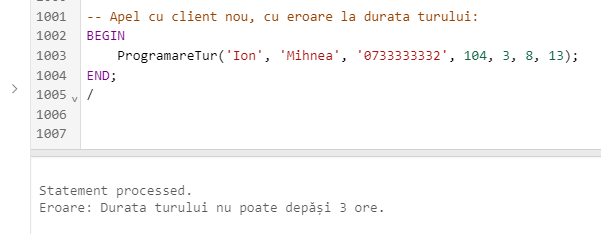
DBMS\_OUTPUT.PUT\_LINE('Eroare necunoscută. ' || SQLCODE);

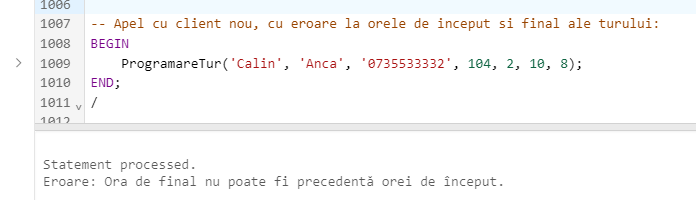
END;

/









1. *Formularea* *enunțului unei probleme rezolvate folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratarea tuturor excepțiilor care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelul subprogramului astfel încât au fot evidențiate toate cazurile tratate.*

Avem un grup de clienti care vor specific sa adopte animalutele cu cea mai mare varsta din centrul cel mai apropiat (se da id-ul centrului).

Sa se creeze un nou contract de adoptie pentru animalutul cu varsta cea mai mare.

Tabele folosite: CLIENT, CONTRACT\_ADOPTIE, CENTRU, ANIMAL\_COMPANIE, FISA\_AC.

CREATE OR REPLACE PROCEDURE Exercitiul9(

p\_nume\_client IN VARCHAR2,

p\_prenume\_client IN VARCHAR2,

p\_telefon\_client IN VARCHAR2,

p\_adresa\_client IN VARCHAR2,

p\_email\_client IN VARCHAR2,

p\_id\_centru IN NUMBER

) AS

v\_nrC CLIENT.nrC%TYPE;

v\_petID ANIMAL\_COMPANIE.petID%TYPE;

v\_id\_contract CONTRACT\_ADOPTIE.id\_contract%TYPE;

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM ANIMAL\_COMPANIE

WHERE id\_centru = p\_id\_centru;

IF v\_count = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu s-au găsit animale de companie în centrul specificat.');

END IF;

SELECT seq\_nrC.NEXTVAL INTO v\_nrC FROM DUAL;

INSERT INTO CLIENT(nrC, nume, prenume, telefon, adresa, email)

VALUES (v\_nrC, p\_nume\_client, p\_prenume\_client, p\_telefon\_client, p\_adresa\_client, p\_email\_client);

SELECT petID INTO v\_petID

FROM (

SELECT ac.petID

FROM ANIMAL\_COMPANIE ac

JOIN FISA\_AC fa ON ac.id\_fisa = fa.id\_fisa -- join pe cheia id\_fisa

WHERE ac.id\_centru = p\_id\_centru

ORDER BY fa.varsta DESC, ac.data\_preluare ASC -- ord după varsta descendent și data\_preluare ascendent (pentru stabilitate)

)

WHERE ROWNUM = 1;

SELECT seq\_id\_contract.NEXTVAL INTO v\_id\_contract FROM DUAL;

INSERT INTO CONTRACT\_ADOPTIE(id\_contract, petID, nrC, data\_adoptie)

VALUES (v\_id\_contract, v\_petID, v\_nrC, SYSDATE);

DBMS\_OUTPUT.PUT\_LINE('Contract de adoptie creat cu succes pentru clientul ' || p\_nume\_client || ' ' || p\_prenume\_client || ' si animalul de companie cu ID-ul ' || v\_petID);

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Prea multe rânduri returnate în subinterogare.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

END;

/

BEGIN

Exercitiul9(

'Popescu',

'Maria',

'0721122334',

'Str. Florilor nr. 10',

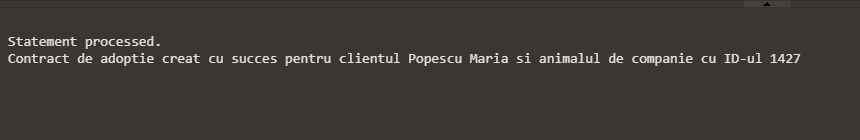
'popescu.maria@example.com',

100

);

END;

/



1. *Definirea și* *declanșarea unui trigger de tip LMD la nivel de comandă.*

CREATE OR REPLACE TRIGGER trg\_LMD\_Donatie

AFTER INSERT OR DELETE ON DONATIE

FOR EACH ROW

ENABLE

DECLARE

v\_op\_tip VARCHAR2(10);

BEGIN

IF INSERTING THEN

v\_op\_tip := 'INSERT';

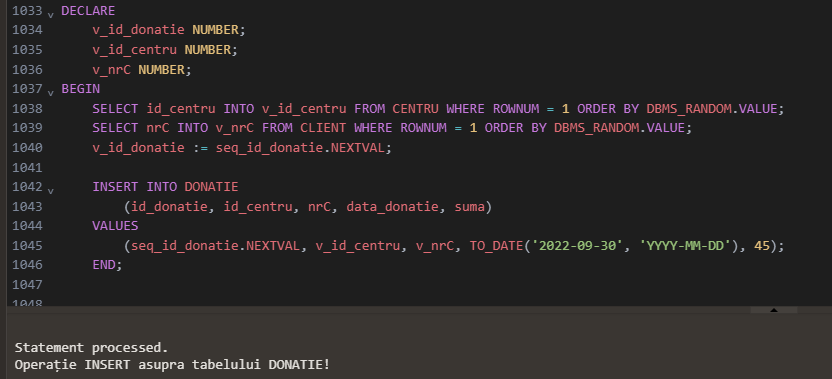
ELSIF DELETING THEN

v\_op\_tip := 'DELETE';

END IF;

DBMS\_OUTPUT.PUT\_LINE('Operație ' || v\_op\_tip || ' asupra donației cu ID: ' || :OLD.id\_donatie);

END;



1. *Definirea și declanșarea unui trigger de tip LMD la nivel de linie.*

CREATE OR REPLACE TRIGGER trg\_LMD\_Donatie

AFTER INSERT OR DELETE ON DONATIE

FOR EACH ROW

ENABLE

DECLARE

v\_op\_tip VARCHAR2(10);

BEGIN

IF INSERTING THEN

v\_op\_tip := 'INSERT';

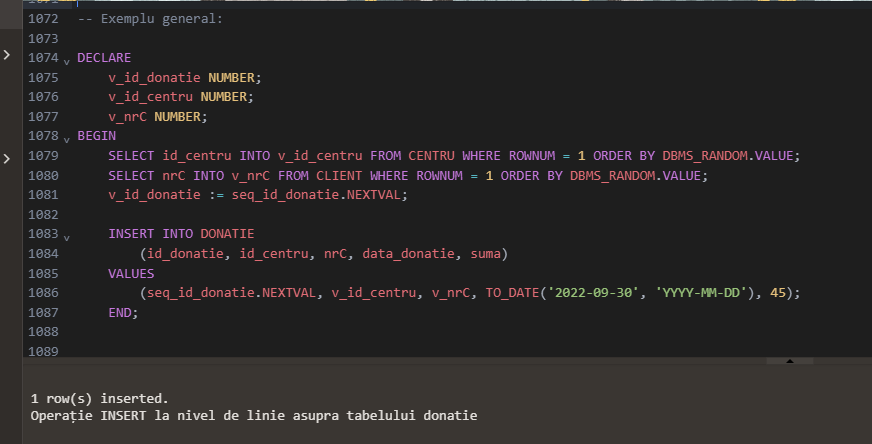
ELSIF DELETING THEN

v\_op\_tip := 'DELETE';

END IF;

DBMS\_OUTPUT.PUT\_LINE('Operație ' || v\_op\_tip || ' la nivel de linie asupra tabelului donatie');

END;



*12. Definirea* *și declanșarea unui trigger de tip LDD.*

CREATE OR REPLACE TRIGGER trg\_LDD\_Donatie

BEFORE DELETE ON DONATIE

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Ștergere donație cu id ' || :OLD.id\_donatie || ' înainte de ștergere.');

END;

