

Proiect SGBD

1. Prezențați pe scurt baza de date (utilitatea ei).

O importantă problemă comunitară, în ziua de astăzi, o constituie gestionarea deșeurilor. Astfel, a apărut necesitatea existenței stațiilor de colectare și sortare de deșeurilor, în vederea reciclării celor care oferă acest beneficiu. Deșeurile pot fi de mai multe tipuri, fiind selectate pe sortimente, culori și compoziție.

Procesul tehnologic constă în sortarea mecanică și manuală a deșeurilor colectate selectiv. Deșeurile sunt balotat pe categorii și livrate centrelor care le reciclează (reprezentând clienții). Furnizorii pot fi atât persoane fizice, cât și juridice: restaurante, hoteluri, magazine, instituții publice etc., dar și coșurile de gunoi din orașe, dotate cu dispozitive inteligente ce anunță centrele de colectare corespunzătoare, când sunt necesare colectările.

Proiectul este o aplicație ce vine în ajutorul proprietarilor de centre de colectare și sortare a deșeurilor. Tabelele conținute sunt:

- tabela **ANGAJATI**: - are PRIMARY KEY câmpul ID.
- tabela **CATEGORII_DESEURI**: - are PRIMARY KEY câmpul ID.
- tabela **CENTRE_DE_COLECTARE**: - are PRIMARY KEY câmpul ID.
- tabela **CLIENTI**: - are PRIMARY KEY câmpul ID.
- tabela **COSURI_DE_GUNOI**: - are PRIMARY KEY câmpul ID_COS.
- tabela **DISPOZITIVE_INTELIGENTE**: - are PRIMARY KEY câmpul ID.
- tabela **FURNIZORI**: - are PRIMARY KEY câmpul ID_FURNIZOR.
- tabela **SOFERI**: - are PRIMARY KEY câmpul ID_SOFER.
- tabela **UTILIZATORI**: - are PRIMARY KEY câmpul ID.
- tabela **VEHICULE**: - are PRIMARY KEY câmpul NUMAR_VEHICUL.

Iar relațiile dintre ele sunt:

Fiecare utilizator poate folosi unul sau mai multe cosuri de gunoi.

Fiecare cos de gunoi poate fi folosit de unul sau mai multi utilizatori.

Fiecare dispozitiv inteligent poate să se afle la un singur cos de gunoi.

Fiecare cos de gunoi trebuie să conțină un singur dispozitiv inteligent.

Fiecare dispozitiv inteligent trebuie să fie monitorizat de un centru de colectare.

Fiecare centru de colectare trebuie să monitorizeze unul sau mai multe dispozitive inteligente.

Fiecare sofer poate fi angajat de un singur centru de colectare.

Fiecare centru de colectare trebuie să angajeze unul sau mai multi soferi.

Fiecare sofer poate sa conduca unul sau mai multe vehicule (nu vorbesc de acelasi moment de timp).

Fiecare vehicul poate fi condus de unul sau mai multi soferi (aceeasi observatie).

Fiecare vehicul trebuie sa fie detinut de un singur centru de colectare.

Fiecare centru de colectare trebuie sa detina unul sau mai multe vehicule.

Fiecare cos de gunoi poate fi colectat de unul sau mai multe vehicule (nu e mandatory, pentru ca pot exista cosuri care sa nu se umple niciodata daca nu sunt accesate de multi utilizator => nu vor trimite nicio alarma catre autoritati).

Fiecare vehicul poate sa se deplaseze la unul sau mai multe cosuri de gunoi.

Fiecare furnizor poate furniza un singur centru de colectare.

Fiecare centru de colectare trebuie sa fie furnizat de unul sau mai multi furnizori.

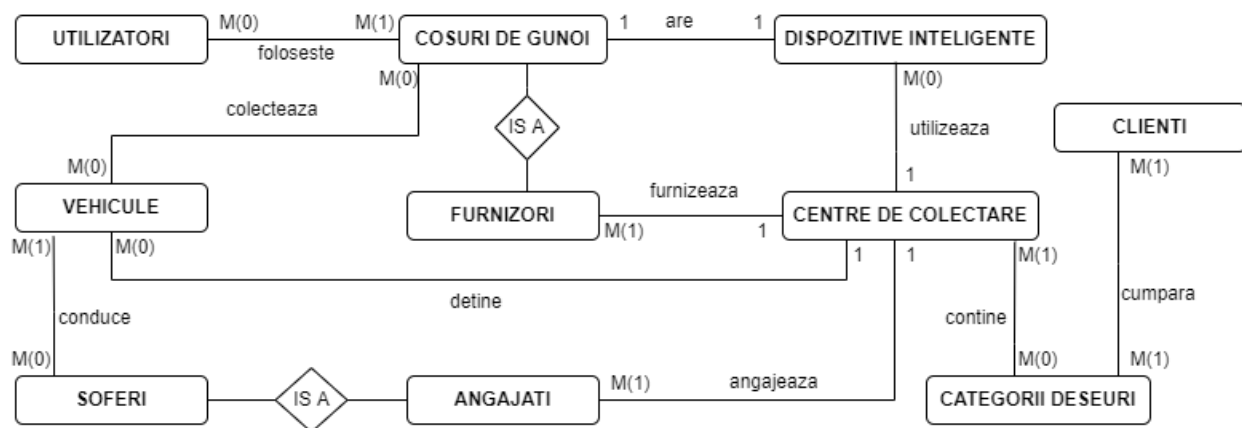
Fiecare centru de colectare trebuie sa contina una sau mai multe categorii de deseuri.

Fiecare categorie de deseuri poate sa fie la unul sau mai multe centre de colectare.

Fiecare categorie de deseuri trebuie sa fie cumparata de unul sau mai multi clienti.

Fiecare client trebuie sa cumpere una sau mai multe categorii de deseuri.

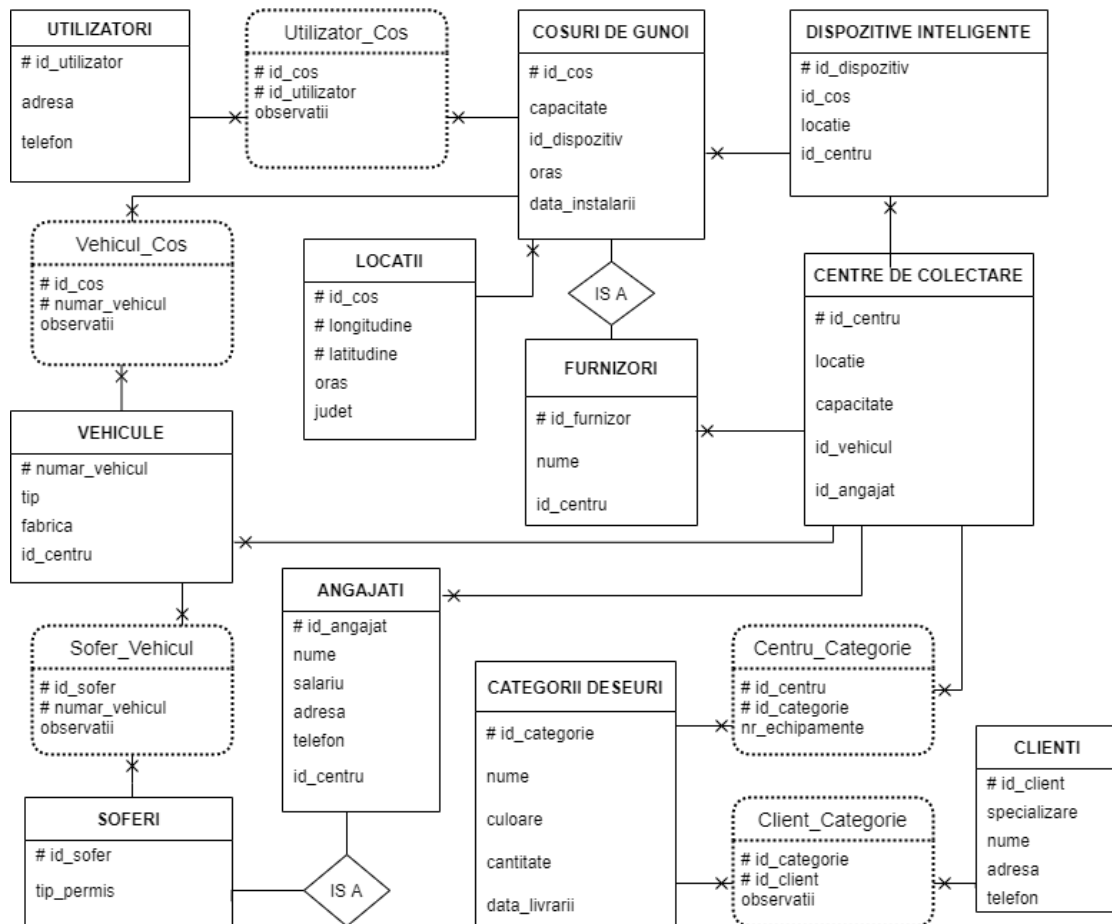
2. Realizați diagrama entitate-relație (ERD).



unde: 1, M = Cardinalitatea maxima

(0), (1) = Cardinalitatea minima => obligativitatea participarii unei entitati la o relatie este totala (1) sau partiala (0)

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



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

```
CREATE TABLE cosuri_de_gunoi(
    id_cos number(4) PRIMARY KEY,    -- coincide cu id_furnizor
    id_dispozitiv number(4) NOT NULL,
    capacitate number(4) NOT NULL,
    oras varchar2(20) NOT NULL,
    data_instalarii date,
    nume varchar2(20) NOT NULL,
    id_centru number(4));
```

```
INSERT INTO cosuri_de_gunoi (id_cos, capacitate, id_dispozitiv, oras, data_instalarii,
nume, id_centru) VALUES (1,7, 1, 'Ploiesti', to_date('02-FEB-14'), 'Cos de gunoi', 3);
```

```

INSERT INTO cosuri_de_gunoi (id_cos, capacitate, id_dispozitiv, oras, data_instalarii,
nume, id_centru) VALUES (2, 3, 2, 'Bacau', to_date('23-SEP-17'), 'Cos de gunoi', 4);

INSERT INTO cosuri_de_gunoi (id_cos, capacitate, id_dispozitiv, oras, data_instalarii,
nume, id_centru) VALUES (3,5,3, 'Sibiu', to_date('12-MAR-15'), 'Cos de gunoi', 5);

INSERT INTO cosuri_de_gunoi (id_cos, capacitate, id_dispozitiv, oras, data_instalarii,
nume, id_centru) VALUES (4,10,4, 'Bucuresti',to_date('08-FEB-14'), 'Cos de gunoi', 2);

INSERT INTO cosuri_de_gunoi (id_cos, capacitate, id_dispozitiv, oras, data_instalarii,
nume, id_centru) VALUES (5,1, 5, 'Craiova', to_date('28-FEB-17'), 'Cos de gunoi', 1);

```

```

-----

CREATE TABLE locatii(
            id_cos number(4) NOT NULL,
            longitudine varchar2(20) NOT NULL,
            latitudine varchar2(20) NOT NULL,
            oras varchar2(20) NOT NULL,
            judet varchar2(20) NOT NULL,
            constraint loc_loc_pk unique (id_cos, longitudine, latitudine));

```

```

INSERT INTO locatii (id_cos, longitudine, latitudine, oras, judet) VALUES (1, '26.03',
'44.94', 'Ploiesti', 'Prahova');

INSERT INTO locatii (id_cos, longitudine, latitudine, oras, judet) VALUES (2, '26.89',
'46.57', 'Bacau', 'Bacau');

INSERT INTO locatii (id_cos, longitudine, latitudine, oras, judet) VALUES (3, '24.16',
'45.78', 'Sibiu', 'Sibiu');

INSERT INTO locatii (id_cos, longitudine, latitudine, oras, judet) VALUES (4, '26.08',
'44.48', 'Bucuresti', 'Bucuresti');

INSERT INTO locatii (id_cos, longitudine, latitudine, oras, judet) VALUES (5, '23.85',
'44.31', 'Craiova', 'Dolj');

```

```

-----

CREATE TABLE centre_de_colectare(
            id_centru number(4) PRIMARY KEY,
            capacitate number(4) NOT NULL,
            locatie varchar2(100) NOT NULL);

alter table centre_de_colectare add (id_angajat number(4));

alter table centre_de_colectare add constraint centru_ang_fk foreign key(id_angajat)
references angajati(id_angajat) on delete set null;

alter table centre_de_colectare add (numar_vehicul varchar2(20));

alter table centre_de_colectare add constraint centru_veh_fk foreign
key(numar_vehicul) references vehicule(numar_vehicul) on delete set null;

-- update centre_de_colectare set id_angajat

```

```

INSERT INTO centre_de_colectare (id_centru, capacitate, locatie) VALUES (1, 14,
'Craiova');
INSERT INTO centre_de_colectare (id_centru, capacitate, locatie) VALUES (2, 45,
'Bucuresti');
INSERT INTO centre_de_colectare (id_centru, capacitate, locatie) VALUES (3, 33,
'Ploiesti');
INSERT INTO centre_de_colectare (id_centru, capacitate, locatie) VALUES (4, 17,
'Bacau');
INSERT INTO centre_de_colectare (id_centru, capacitate, locatie) VALUES (5, 25,
'Sibiu');
INSERT INTO centre_de_colectare (id_centru, capacitate, locatie) VALUES (6, 30,
'Timisoara');

```

```

-----
CREATE TABLE furnizori(
            id_furnizor number(4) PRIMARY KEY,
            nume varchar2(20) NOT NULL,
            id_centru number(4),
            constraint furnizori_centru_fk foreign key (id_centru)
            references centre_de_colectare(id_centru)
            on delete set null);

```

```

INSERT INTO furnizori (id_furnizor, nume, id_centru) VALUES (1, 'Cos de gunoi', 3);
INSERT INTO furnizori (id_furnizor, nume, id_centru) VALUES (2, 'Cos de gunoi', 4);
INSERT INTO furnizori (id_furnizor, nume, id_centru) VALUES (3, 'Cos de gunoi', 5);
INSERT INTO furnizori (id_furnizor, nume, id_centru) VALUES (4, 'Cos de gunoi', 2);
INSERT INTO furnizori (id_furnizor, nume, id_centru) VALUES (5, 'Cos de gunoi', 1);
INSERT INTO furnizori (id_furnizor, nume, id_centru) VALUES (6, 'Carrefour Blejoi',3);
INSERT INTO furnizori (id_furnizor, nume, id_centru) VALUES (7, 'Parc Herastrau', 2);

```

```

-----
CREATE TABLE categorii_deseuri(
            id_categorie number(4) PRIMARY KEY,
            nume varchar2(20) NOT NULL,
            culoare varchar2(20),
            cantitate number(4) NOT NULL,
            data_livrarii date);

```

```

INSERT INTO categorii_deseuri (id_categorie, nume, culoare, cantitate, data_livrarii)
VALUES (1, 'Plastic', 'Galben', 8, to_date('23-FEB-17'));

INSERT INTO categorii_deseuri (id_categorie, nume, culoare, cantitate, data_livrarii)
VALUES (2, 'Hartie', 'Albastru', 5, to_date('03-MAR-17'));

INSERT INTO categorii_deseuri (id_categorie, nume, culoare, cantitate, data_livrarii)
VALUES (3, 'Deseuri menajere', 'Maro', 20, to_date('29-SEP-16'));

INSERT INTO categorii_deseuri (id_categorie, nume, culoare, cantitate, data_livrarii)
VALUES (4, 'Sticlă', 'Verde', 16, to_date('12-FEB-14'));

INSERT INTO categorii_deseuri (id_categorie, nume, culoare, cantitate, data_livrarii)
VALUES (5, 'Metal', 'Rosu', 11, to_date('08-OCT-16'));

```

```

-----
CREATE TABLE centru_categorie (
        id_centru number(4) NOT NULL,
        id_categorie number(4) NOT NULL,
        nr_echipamente number(4),
        constraint cc_cc_pk unique(id_centru, id_categorie));

```

```

INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (1,1,4);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (2,5,7);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (5,3,1);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (4,5,9);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (1,2,5);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (3,2,4);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (3,4,11);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (4,1,3);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (2,3,6);
INSERT INTO centru_categorie (id_centru, id_categorie, nr_echipamente) VALUES (5,4,2);

```

```

-----
CREATE TABLE clienti (
        id_client number(4) PRIMARY KEY,
        nume varchar2(20) NOT NULL,
        specializare varchar2(100) NOT NULL,
        telefon number(10) NOT NULL,
        adresa varchar2(100) NOT NULL );

```

```

INSERT INTO clienti (id_client, nume, specializare, telefon, adresa) VALUES (1, 'Pacos
Eco', 'Plastic, Metal', 0709882991, 'Bacau');

```

```

INSERT INTO clienti (id_client, nume, specializare, telefon, adresa) VALUES (2, 'Remat
PH', 'Metal, Sticla', 0299782911, 'Bucuresti');
INSERT INTO clienti (id_client, nume, specializare, telefon, adresa) VALUES (3,
'Stilos Com', 'Sticla, Metal, Hartie', 0791188551, 'Blejoii');
INSERT INTO clienti (id_client, nume, specializare, telefon, adresa) VALUES (4,
'Intercom TT', 'Deseuri menajere, Plastic', 0779783921, 'Sibiu');
INSERT INTO clienti (id_client, nume, specializare, telefon, adresa) VALUES (5, 'Nevo
SRL', 'Hartie, Metal', 0229282929, 'Craiova');

```

```

-----
CREATE TABLE client_categorie (
    id_client number(4) NOT NULL,
    id_categorie number(4) NOT NULL,
    observatii varchar2(20),
    constraint clc_clc_pk unique(id_client, id_categorie));

INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(1,1,'plastic');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(4,1,'plastic');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(3,2,'hartie');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(5,2,'hartie');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(4,3,'deseuri
mesajere');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(2,4,'sticla');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(3,4,'sticla');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(1,5,'metal');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(2,5,'metal');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(3,5,'metal');
INSERT INTO client_categorie(id_client,id_categorie,observatii) VALUES(5,5,'metal');

```

```

-----
CREATE TABLE dispozitive_inteligente(
    id_dispozitiv number(4) PRIMARY KEY,
    id_cos number(4),
    locatie varchar2(100),
    id_centru number(4),
    constraint disp_cos_fk foreign key (id_cos)
    references cosuri_de_gunoii(id_cos)
    on delete set null,
    constraint disp_centru_fk foreign key (id_centru)

```

```
references centre_de_colectare(id_centru)
on delete set null);
```

```
INSERT INTO dispozitive_inteligente (id_dispozitiv, id_cos, locatie, id_centru) VALUES
(1, 5, 'Craiova', 1);
INSERT INTO dispozitive_inteligente (id_dispozitiv, id_cos, locatie, id_centru) VALUES
(2, 2, 'Bacau', 4);
INSERT INTO dispozitive_inteligente (id_dispozitiv, id_cos, locatie, id_centru) VALUES
(3, 1, 'Ploiesti', 3);
INSERT INTO dispozitive_inteligente (id_dispozitiv, id_cos, locatie, id_centru) VALUES
(4, 3, 'Sibiu', 5);
INSERT INTO dispozitive_inteligente (id_dispozitiv, id_cos, locatie, id_centru) VALUES
(5, 4, 'Bucuresti', 2);
```

```
-----
CREATE TABLE angajati(
    id_angajat number(4) PRIMARY KEY,
    nume varchar2(20) NOT NULL,
    salariu number(10),
    telefon number(10) NOT NULL,
    adresa varchar2(100),
    id_centru number(4) NOT NULL,
    constraint ang_centru_fk foreign key (id_centru)
    references centre_de_colectare(id_centru)
    on delete set null);
```

```
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru) VALUES
(1, 'Popescu Andrei', 2500, 0712479287, 'Ploiesti', 3);
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru) VALUES
(2, 'Ionescu Stefan', 1800, 0723579286, 'Bucuresti', 2);
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru) VALUES
(3, 'Stan Victor', 2570, 0702742881, 'Sibiu', 5);
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru) VALUES
(4, 'Radu Aiana', 1500, 0777679222, 'Craiova', 1);
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru) VALUES
(5, 'Albu Maria', 3200, 0788499350, 'Bacau', 4);
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru) VALUES
(6, 'Gheorghe Ana', 5000, 072494555, 'Bucuresti', 2);
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru) VALUES
(7, 'Visan Robert', 2200, 0798699324, 'Ploiesti', 3);
```

```
-----
```



```

CREATE TABLE soferi(
    id_sofer number(4) PRIMARY KEY, -- coincide cu id_angajat
    tip_permis varchar2(20),
    nume varchar2(20) NOT NULL,
    salariu number(10),
    telefon number(10),
    adresa varchar2(100),
    id_centru number(4) NOT NULL,
    constraint soferi_centru_fk foreign key (id_centru)
    references centre_de_colectare(id_centru)
    on delete set null);

INSERT INTO soferi (id_sofer, tip_permis, nume, salariu, telefon, adresa, id_centru)
VALUES (1, 'B', 'Popescu Andrei', 2500, 0712479287, 'Ploiesti', 3); -- autovehicul
tragator(<3500kg) + remorca(<750kg)

INSERT INTO soferi (id_sofer, tip_permis, nume, salariu, telefon, adresa, id_centru)
VALUES (2, 'D', 'Ionescu Stefan', 1800, 0723579286, 'Bucuresti', 2); -- transport
persoane

INSERT INTO soferi (id_sofer, tip_permis, nume, salariu, telefon, adresa, id_centru)
VALUES (3, 'DE', 'Stan Victor', 2570, 0702742881, 'Sibiu', 5); -- autovehicul
tragator + remorca(>750kg)

INSERT INTO soferi (id_sofer, tip_permis, nume, salariu, telefon, adresa, id_centru)
VALUES (4, 'Tr', 'Radu Aiana', 1500, 0777679222, 'Craiova', 1); -- tractoare
agricole/forestiere

INSERT INTO soferi (id_sofer, tip_permis, nume, salariu, telefon, adresa, id_centru)
VALUES (5, 'CE', 'Albu Maria', 3200, 0788499350, 'Bacau', 4); -- autovehicul
tragator(>3500kg) + (semi)remorca (>750kg)

INSERT INTO soferi (id_sofer, tip_permis, nume, salariu, telefon, adresa, id_centru)
VALUES (6, 'B', 'No Vehicul s', 3200, 0788499350, 'Bucuresti', 3);

```

```

-----
CREATE TABLE vehicule(
    numar_vehicul varchar2(20) PRIMARY KEY,
    tip varchar2(20),
    fabrica varchar2(20),
    id_centru number(4) NOT NULL,
    constraint veh_centru_fk foreign key (id_centru)
    references centre_de_colectare(id_centru)
    on delete set null);

```

```

INSERT INTO vehicule (numar_vehicul, tip, fabrica, id_centru) VALUES ('PH 76 ABC',
'Gunoiera', 'Renault', 3);

```

```

INSERT INTO vehicule (numar_vehicul, tip, fabrica, id_centru) VALUES ('B 336 NNC',
'Camion', 'Volkswagen', 2);
INSERT INTO vehicule (numar_vehicul, tip, fabrica, id_centru) VALUES ('SB 12 KQM',
'Gunoiera', 'Ford', 5);
INSERT INTO vehicule (numar_vehicul, tip, fabrica, id_centru) VALUES ('DJ 94 ABB',
'Autogunoiera', 'Norba', 1);
INSERT INTO vehicule (numar_vehicul, tip, fabrica, id_centru) VALUES ('BC 33 STI',
'Camion', 'Ford', 4);

```

```

-----
CREATE TABLE sofer_vehicul(
            id_sofer number(4) NOT NULL,
            numar_vehicul varchar2(20) NOT NULL,
            observatii varchar2(20),
            constraint sof_veh_pk unique(id_sofer, numar_vehicul));

```

```

INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(4,'DJ 94 ABB','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(1,'PH 76 ABC','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(2,'B 336 NNC','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(3,'SB 12 KQM','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(5,'B 336 NNC','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(5,'BC 33 STI','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(1,'B 336 NNC','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(4,'B 336 NNC','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(1,'BC 33 STI','');
INSERT INTO sofer_vehicul(id_sofer,numar_vehicul,observatii) VALUES(2,'PH 76 ABC','');

```

```

-----
CREATE TABLE vehicul_cos (
            id_cos number(4) NOT NULL,
            numar_vehicul varchar2(20) NOT NULL,
            observatii varchar2(20),
            constraint veh_cos_pk unique (id_cos, numar_vehicul));

```

```

INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (1, 'PH 76 ABC', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (2, 'BC 33 STI', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (3, 'SB 12 KQM', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (4, 'B 336 NNC', '');

```

```

INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (5, 'DJ 94 ABB', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (1, 'B 336 NNC', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (4, 'PH 76 ABC', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (5, 'B 336 NNC', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (3, 'PH 76 ABC', '');
INSERT INTO vehicul_cos (id_cos,numar_vehicul,observatii) VALUES (1, 'BC 33 STI', '');

```

```

-----
CREATE TABLE utilizatori(
            id_utilizator number(4) PRIMARY KEY,
            telefon number(10),
            adresa varchar2(100));

```

```

CREATE SEQUENCE s INCREMENT BY 1 START WITH 1;
INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval,
0733195500, 'Bacau Str. Democratiei');
INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval,
0788499350, 'Baicoi Str. Mihai Eminescu');
INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval,
0701239973, 'Sibiu Str. Cernei');
INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval,
0722451306, 'Bucuresti Bd Independentei');
INSERT INTO utilizatori (id_utilizator, telefon, adresa) VALUES (s.nextval,
0714136909, 'Craiova Str. Zorilor');

```

```

-----
CREATE TABLE utilizator_cos(
            id_cos number(4) NOT NULL,
            id_utilizator number(4) NOT NULL,
            observatii varchar2(100),
            constraint utiliz_cos_pk unique(id_cos, id_utilizator));

```

```

INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (1, 2,
'Ploiesti-Baicoi');
INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (1, 4,
'Ploiesti-Bucuresti');
INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (1, 1,
'Ploiesti-Bacau');
INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (2, 1, 'Bacau-
Bacau');

```

```

INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (2, 2, 'Bacau-
Baicoi');

INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (3, 3, 'Sibiu-
Sibiu');

INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (3, 4, 'Sibiu-
Bucuresti');

INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (4, 5,
'Bucuresti-Craiova');

INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (4, 4,
'Bucuresti-Bucuresti');

INSERT INTO utilizator_cos (id_cos, id_utilizator, observatii) VALUES (5, 5, 'Craiova-
Craiova');

```

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

The screenshot shows a database management tool interface with a 'Script Output' window. The window displays the results of a script execution, including table creation and data insertion. The output is organized into two columns, with a 'Task completed in 0.26 seconds' status bar at the top.

Script Output	Task completed in 0.26 seconds
Table LOCATII created.	Table FURNIZORI created.
1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.
Table CENTRE_DE_COLECTARE created.	1 row inserted.
Table CENTRE_DE_COLECTARE altered.	1 row inserted.
	Table CATEGORII_DESEURI created.

At the bottom of the window, there is a 'Messages - Log' section with tabs for 'Messages', 'Logging Page', and 'Statements'.

Table CENTRU_CATEGORIE created.	Table CLIENTI created.	Table DISPOZITIVE_INTELIGENTE created
1 row inserted.	1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.	1 row inserted.
1 row inserted.	1 row inserted.	1 row inserted.
1 row inserted.	Table CLIENT_CATEGORIE created.	Table ANGAJATI created.
1 row inserted.	1 row inserted.	1 row inserted.

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.

```
-- Afisati toti clientii unui centru de colectare cu locatia data ca parametru si
cosurile de gunoi (id-urile) de la locatia respectiva.
-- => colectii de tip tablou indexat si vector.
CREATE OR REPLACE PROCEDURE cerinta6 (loc centre_de_colectare.locatie%TYPE) AS
    TYPE tablou IS TABLE OF clienti%ROWTYPE INDEX BY PLS_INTEGER;
    t tablou;
    type vector is varray(20) of number(3); -- va contine id-urile cosurilor de gunoi
    v vector := vector();
BEGIN
    SELECT id_cos BULK COLLECT INTO v
    FROM cosuri_de_gunoi
    WHERE oras = loc;

    DBMS_OUTPUT.PUT_LINE('Exista ' || v.COUNT || ' cosuri de gunoi pentru locatia ' ||
    loc);

    SELECT * BULK COLLECT INTO t
    FROM clienti
    WHERE adresa = loc;
```

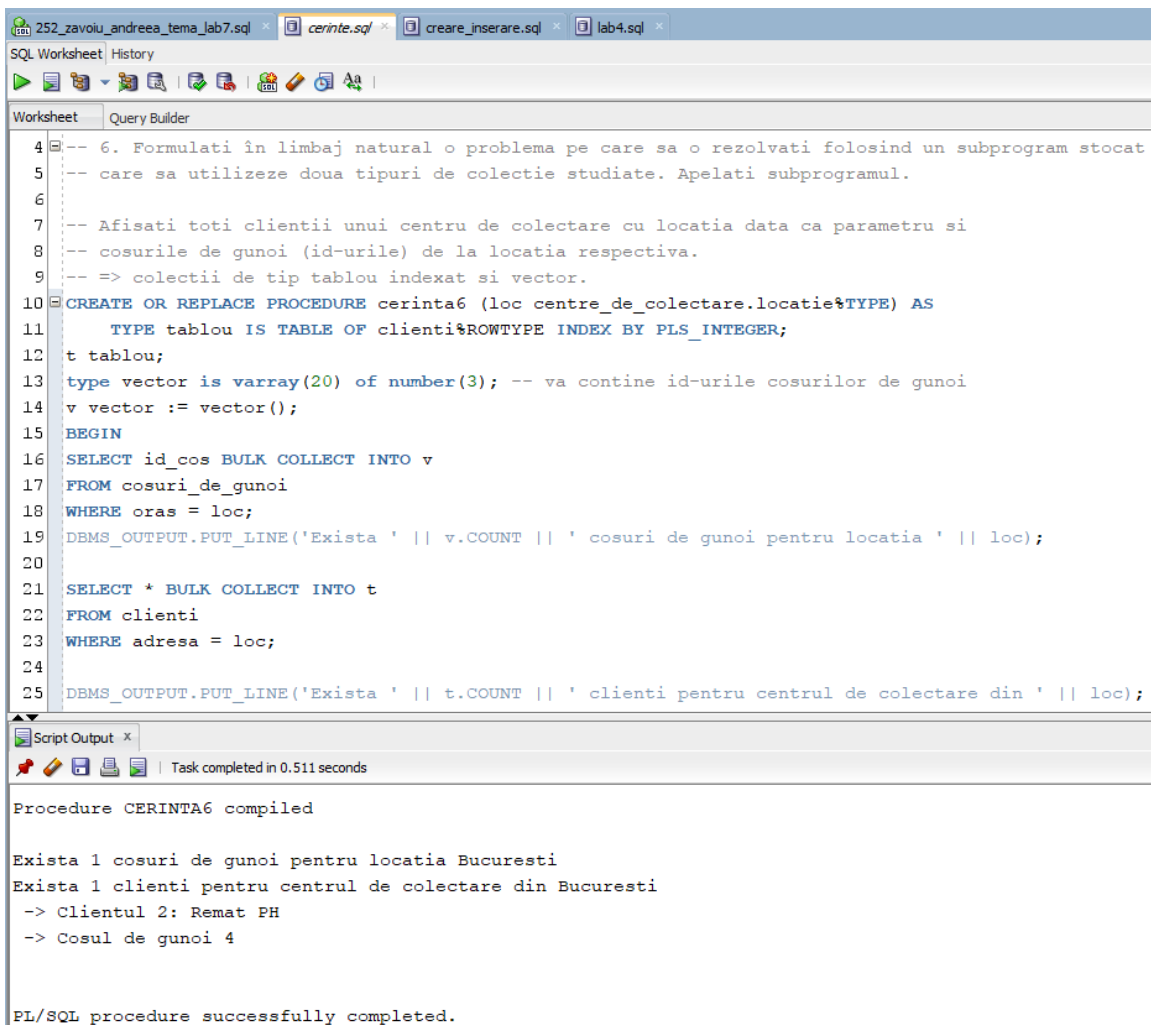
```

DBMS_OUTPUT.PUT_LINE('Exista ' || t.COUNT || ' clienti pentru centrul de colectare
din ' || loc);

FOR i IN t.FIRST..t.LAST LOOP
    DBMS_OUTPUT.PUT_LINE(' -> Clientul ' || t(i).id_client || ': ' || t(i).nume);
END LOOP;

FOR j IN v.FIRST..v.LAST LOOP
    DBMS_OUTPUT.PUT_LINE(' -> Cosul de gunoi ' || v(j));
END LOOP;
END;
/
BEGIN
    cerinta6('Bucuresti'); -- locatia data
END;
/

```



The screenshot shows an SQL Worksheet application with multiple tabs. The active tab is 'cerinta6.sql'. The code in the worksheet is a PL/SQL procedure named 'cerinta6' that takes a location as input. It uses two collections: a table 't' for clients and a vector 'v' for waste bins. The procedure queries the database for clients at the specified location and waste bins at that location, then prints the results. The 'Script Output' window at the bottom shows the execution results, indicating that the procedure was compiled successfully and the output matches the expected results from the code above.

```

252_zavoiu_andreea_tema_lab7.sql x cerinta6.sql x creare_inserare.sql x lab4.sql x
SQL Worksheet History
Worksheet Query Builder
4 -- 6. Formulati în limbaj natural o problema pe care sa o rezolvati folosind un subprogram stocat
5 -- care sa utilizeze doua tipuri de colectie studiate. Apelati subprogramul.
6
7 -- Afisati toti clientii unui centru de colectare cu locatia data ca parametru si
8 -- cosurile de gunoi (id-urile) de la locatia respectiva.
9 -- => colectii de tip tablou indexat si vector.
10 CREATE OR REPLACE PROCEDURE cerinta6 (loc centre_de_colectare.locatie%TYPE) AS
11     TYPE tablou IS TABLE OF clienti%ROWTYPE INDEX BY PLS_INTEGER;
12     t tablou;
13     type vector is varray(20) of number(3); -- va contine id-urile cosurilor de gunoi
14     v vector := vector();
15 BEGIN
16     SELECT id_cos BULK COLLECT INTO v
17     FROM cosuri_de_gunoi
18     WHERE oras = loc;
19     DBMS_OUTPUT.PUT_LINE('Exista ' || v.COUNT || ' cosuri de gunoi pentru locatia ' || loc);
20
21     SELECT * BULK COLLECT INTO t
22     FROM clienti
23     WHERE adresa = loc;
24
25     DBMS_OUTPUT.PUT_LINE('Exista ' || t.COUNT || ' clienti pentru centrul de colectare din ' || loc);
Script Output x
Task completed in 0.511 seconds

Procedure CERINTA6 compiled

Exista 1 cosuri de gunoi pentru locatia Bucuresti
Exista 1 clienti pentru centrul de colectare din Bucuresti
-> Clientul 2: Remat PH
-> Cosul de gunoi 4

PL/SQL procedure successfully completed.

```

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.

```
-- Afisati pentru fiecare centru de colectare cati angajati lucreaza acolo. Numele
-- centrelor si numarul de angajati pentru fiecare sunt salvate intr-un cursor.
CREATE OR REPLACE PROCEDURE cerinta7 AS v_centru centre_de_colectare.id_centru%TYPE;
v_nr NUMBER(4);
CURSOR c IS SELECT c.id_centru, COUNT(a.id_angajat) nrAng
              FROM angajati a RIGHT JOIN centre_de_colectare c ON (c.id_centru =
a.id_centru)
              GROUP BY c.id_centru
              ORDER BY c.id_centru;
BEGIN
    OPEN c;
    LOOP
        FETCH c INTO v_centru, v_nr;
        EXIT WHEN c%NOTFOUND;
        IF v_nr = 0 THEN DBMS_OUTPUT.PUT_LINE('La centrul ' || v_centru || ' nu
lucreaza angajati.');
```

ELSIF v_nr = 1 THEN DBMS_OUTPUT.PUT_LINE('La centrul ' || v_centru || '
lucreaza un angajat.');

ELSE DBMS_OUTPUT.PUT_LINE('La centrul ' || v_centru || ' lucreaza ' || v_nr ||
angajati.');

END IF;

END LOOP;

CLOSE c;

END;

/

BEGIN

cerinta7();

END;

/

```

55 CREATE OR REPLACE PROCEDURE cerinta7 AS v_centru centre_de_colectare.id_centru%TYPE;
56 v_nr NUMBER(4);
57 CURSOR c IS SELECT c.id_centru, COUNT(a.id_angajat) nrAng
58 FROM angajati a RIGHT JOIN centre_de_colectare c ON (c.id_centru = a.id_centru)
59 GROUP BY c.id_centru
60 ORDER BY c.id_centru;
61 BEGIN
62 OPEN c;
63 LOOP
64 FETCH c INTO v_centru, v_nr;
65 EXIT WHEN c%NOTFOUND;
66 IF v_nr = 0 THEN DBMS_OUTPUT.PUT_LINE('La centrul ' || v_centru || ' nu lucreaza angajati.');
```

```

La centrul 1 lucreaza un angajat.
La centrul 2 lucreaza 2 angajati.
La centrul 3 lucreaza 2 angajati.
La centrul 4 lucreaza un angajat.
La centrul 5 lucreaza un angajat.
La centrul 6 nu lucreaza angajati.

PL/SQL procedure successfully completed.

```

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.

-- Sa se afiseze angajatii care sunt soferi si stau in unul din orasele in care se afla si centrul de colectare dat.

```

CREATE OR REPLACE FUNCTION cerinta8 (centrulDat centre_de_colectare.id_centru%type)
RETURN VARCHAR2
IS
    TYPE rec IS RECORD (nrAng NUMBER, v_nume VARCHAR2(30), v_adresa VARCHAR2(30));
    ang rec;

```



```

TYPE vector IS VARRAY(20) OF centre_de_colectare.id_centru%TYPE;
v vector; -- neinitializat, adica fara :=vector();
BEGIN
    -- am vrut sa evidentiez si folosirea exceptiei collection_is_null, asa ca am
    facut un if special care sa o apeleze pentru inputul 0
    IF centrulDat = 0 THEN v(1) := centrulDat; END IF;

    SELECT COUNT(id_angajat), nume, adresa INTO ang
    FROM angajati a                                -- primul tabel
    WHERE id_centru = centrulDat AND id_centru in
        (SELECT id_centru FROM soferi                -- al doilea
         WHERE nume = a.nume and adresa in (SELECT locatie FROM
centre_de_colectare))                               -- al treilea
    GROUP BY nume, adresa
    ORDER BY id_centru;

    RETURN ('Pentru centrul ' || centrulDat || ' avem ' || ang.nrAng || ' angajati
soferi.');
```

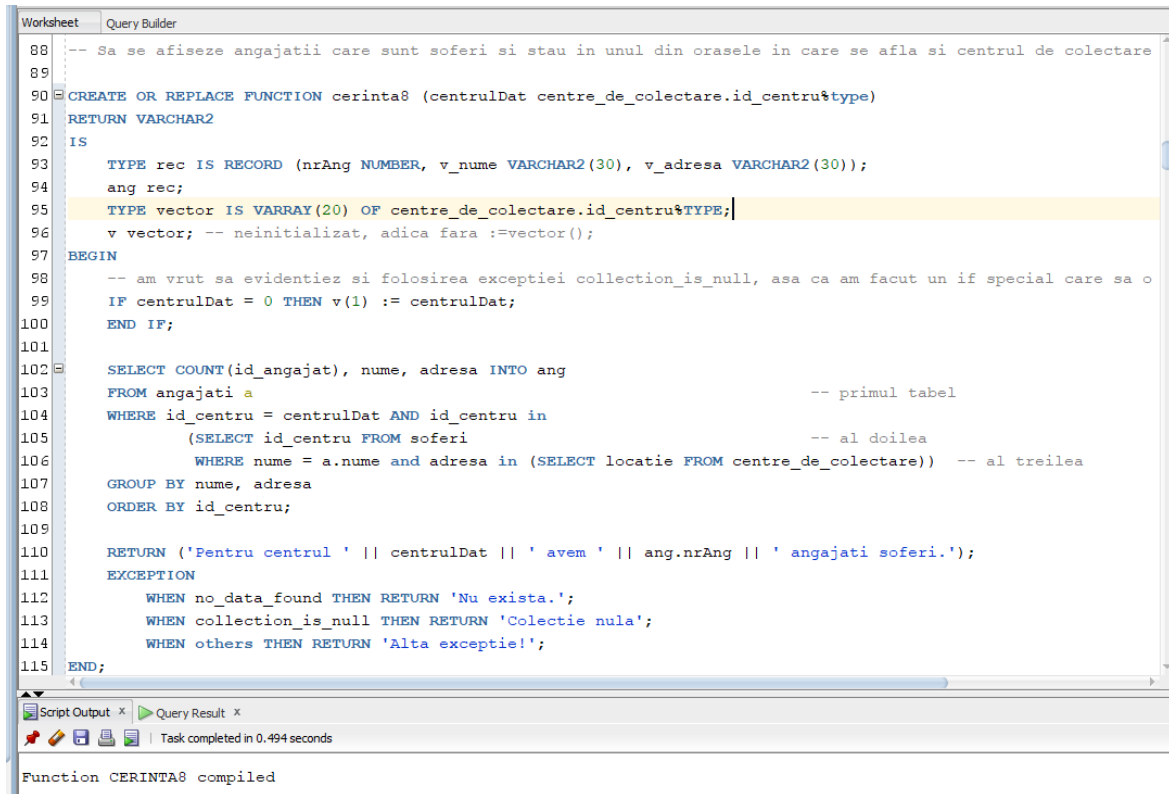
```

EXCEPTION
    WHEN no_data_found THEN RETURN 'Nu exista.';
    WHEN collection_is_null THEN RETURN 'Colectie nula';
    WHEN others THEN RETURN 'Alta exceptie!';
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('-> Pentru id_centru inexistent:');
    DBMS_OUTPUT.PUT_LINE(cerinta8(70)); -- centru fara angajati / neinreg in BD
END;
BEGIN
    DBMS_OUTPUT.PUT_LINE('-> Pentru collection_is_null:');
    DBMS_OUTPUT.PUT_LINE(cerinta8(0)); -- if special pt 0
END;
BEGIN
    DBMS_OUTPUT.PUT_LINE('-> Pentru centru cu angajati soferi (deci nicio eroare
aruncata:');
    DBMS_OUTPUT.PUT_LINE(cerinta8(2)); -- merge cu codurile 1-5
END;
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('-> Pentru centru fara angajati:');
```

```
DBMS_OUTPUT.PUT_LINE(cerinta8(6));
```

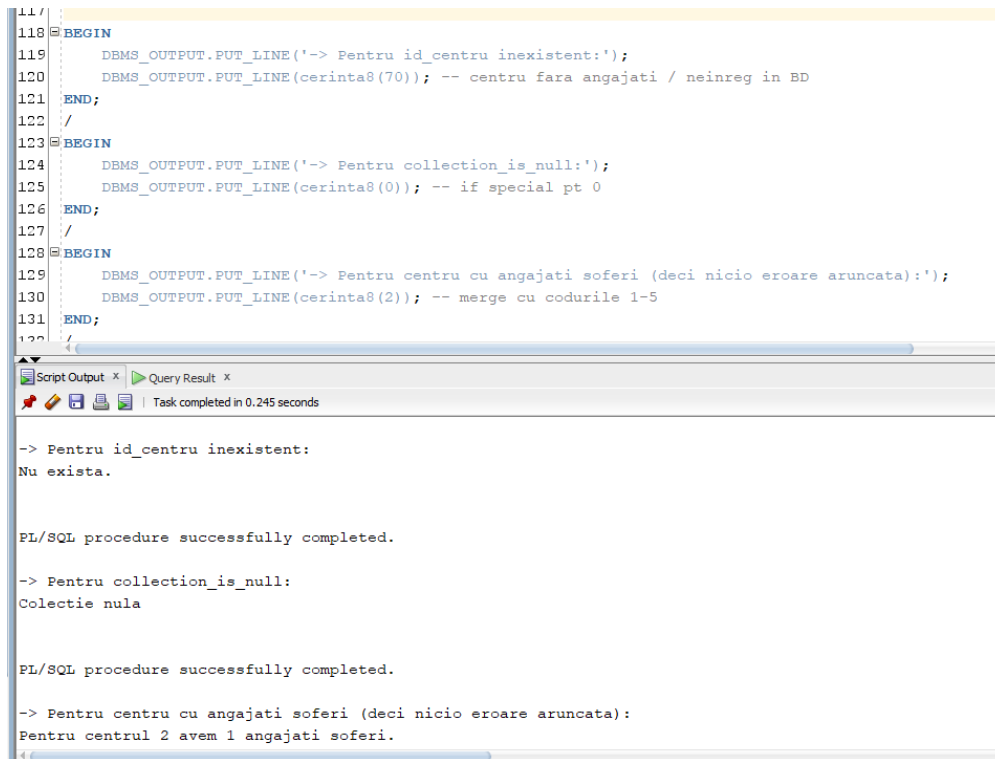
```
END;
```



The screenshot shows the SQL Developer interface with a worksheet containing the following PL/SQL code:

```
88 -- Sa se afiseze angajatii care sunt soferi si stau in unul din orasele in care se afla si centrul de colectare
89
90 CREATE OR REPLACE FUNCTION cerinta8 (centrulDat centre_de_colectare.id_centru%TYPE)
91 RETURN VARCHAR2
92 IS
93     TYPE rec IS RECORD (nrAng NUMBER, v_nume VARCHAR2(30), v_adresa VARCHAR2(30));
94     ang rec;
95     TYPE vector IS VARRAY(20) OF centre_de_colectare.id_centru%TYPE;
96     v vector; -- neinitializat, adica fara :=vector();
97 BEGIN
98     -- am vrut sa evidentiez si folosirea exceptiei collection_is_null, asa ca am facut un if special care sa o
99     IF centrulDat = 0 THEN v(1) := centrulDat;
100     END IF;
101
102     SELECT COUNT(id_angajat), nume, adresa INTO ang
103     FROM angajati a
104     WHERE id_centru = centrulDat AND id_centru in
105           (SELECT id_centru FROM soferi
106            WHERE nume = a.nume and adresa in (SELECT locatie FROM centre_de_colectare))
107     GROUP BY nume, adresa
108     ORDER BY id_centru;
109
110     RETURN ('Pentru centrul ' || centrulDat || ' avem ' || ang.nrAng || ' angajati soferi.');
```

The code continues with exception handling and ends with the function definition. The status bar indicates the task was completed in 0.494 seconds.



The screenshot shows the execution of the PL/SQL function CERINTA8 with the following code:

```
111 /
112
113 BEGIN
114     DBMS_OUTPUT.PUT_LINE('-> Pentru id_centru inexistent:');
115     DBMS_OUTPUT.PUT_LINE(cerinta8(70)); -- centru fara angajati / neinreg in BD
116 END;
117 /
118
119 BEGIN
120     DBMS_OUTPUT.PUT_LINE('-> Pentru collection_is_null:');
121     DBMS_OUTPUT.PUT_LINE(cerinta8(0)); -- if special pt 0
122 END;
123 /
124
125 BEGIN
126     DBMS_OUTPUT.PUT_LINE('-> Pentru centru cu angajati soferi (deci nicio eroare aruncata):');
127     DBMS_OUTPUT.PUT_LINE(cerinta8(2)); -- merge cu codurile 1-5
128 END;
129 /
```

The status bar indicates the task was completed in 0.245 seconds.

The output of the function execution is shown below:

```
-> Pentru id_centru inexistent:
Nu exista.

PL/SQL procedure successfully completed.

-> Pentru collection_is_null:
Colectie nula

PL/SQL procedure successfully completed.

-> Pentru centru cu angajati soferi (deci nicio eroare aruncata):
Pentru centrul 2 avem 1 angajati soferi.
```

```

133 BEGIN
134     DBMS_OUTPUT.PUT_LINE('-> Pentru centru fara angajati:');
135     DBMS_OUTPUT.PUT_LINE(cerinta8(6));
136 END;
137 /
138

```

Script Output x Query Result x

Task completed in 0.174 seconds

```

-> Pentru centru fara angajati:
Nu exista.

PL/SQL procedure successfully completed.

```

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.

-- Procedura care returneaza clientul categoriei de deseuri ce provine de la cosul de gunoi aflat in judetul dat ca parametru de intrare.

```

CREATE OR REPLACE PROCEDURE cerinta9
    (v_judet IN locatii.judet%TYPE,
     raspuns OUT clienti%ROWTYPE) IS
BEGIN
    select * into raspuns from clienti where id_client in (
        select id_client from client_categorie where id_categorie in (
            select id_categorie from centru_categorie where id_centru in (
                select id_centru from cosuri_de_gunoi where id_cos in (
                    select id_cos from locatii where lower(judet)=lower(v_judet))));
    EXCEPTION
        WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20000, 'Nu exista clienti');
        WHEN TOO_MANY_ROWS THEN RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi
        clienti');
        WHEN OTHERS THEN RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');

END;
/

DECLARE output clienti%ROWTYPE;
BEGIN
    cerinta9('Arad', output);

```

```
DBMS_OUTPUT.PUT_LINE('Cerinta9: clientul cu id-ul ' || output.id_client || ' si  
numele ' || output.nume);
```

```
END;
```

```
/
```

```
DECLARE output clienti%ROWTYPE;
```

```
BEGIN cerinta9('Prahova', output); -- exista mai multi clienti
```

```
END;
```

```
/
```

```
DECLARE output clienti%ROWTYPE;
```

```
BEGIN cerinta9('Iasi', output); -- nu exista clienti
```

```
END;
```

```
/
```

```
184 -- Procedura care returneaza clientul categoriei de deseuri ce provine de la cosul de gunoi  
185 -- aflat in judetul dat ca parametru de intrare.  
186  
187 CREATE OR REPLACE PROCEDURE cerinta9  
188     (v_judet IN locatii.judet%TYPE,  
189      raspuns OUT clienti%ROWTYPE) IS  
190 BEGIN  
191     select * into raspuns  
192     from clienti where id_client in (  
193         select id_client from client_categorii where id_categorie in (  
194             select id_categorie from centru_categorii where id_centru in (  
195                 select id_centru from cosuri_de_gunoi where id_cos in (  
196                     select id_cos from locatii where lower(judet) = lower(v_judet))));  
197  
198     EXCEPTION  
199         WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20000, 'Nu exista clienti');  
200         WHEN TOO_MANY_ROWS THEN RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi clienti');  
201         WHEN OTHERS THEN RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');  
202 END;  
203 /  
204
```

Script Output x

Task completed in 0.579 seconds

Procedure CERINTA9 compiled

```

222 DECLARE
223     output clienti%ROWTYPE;
224 BEGIN
225     cerinta9('Arad', output);
226     DBMS_OUTPUT.PUT_LINE('Cerinta9: clientul cu id-ul ' || output.id_client || ' si numele ' ||
227 END;
228 /
229
230 DECLARE output clienti%ROWTYPE;
231 BEGIN cerinta9('Prahova', output); -- exista mai multi clienti
232 END;
233 /

```

Script Output x

Task completed in 0.294 seconds

Cerinta9: clientul cu id-ul 6 si numele Cleanest Clean

PL/SQL procedure successfully completed.

Error starting at line : 230 in command -
 DECLARE output clienti%ROWTYPE;
 BEGIN cerinta9('Prahova', output); -- exista mai multi clienti
 END;
 Error report -
 ORA-20001: Exista mai multi clienti
 ORA-06512: at "ADMIN.CERINTA9", line 16
 ORA-06512: at line 2

```

234
235 DECLARE output clienti%ROWTYPE;
236 BEGIN cerinta9('Iasi', output); -- nu exista clienti
237 END;
238 /
239

```

Script Output x

Task completed in 0.28 seconds

Error starting at line : 235 in command -
 DECLARE output clienti%ROWTYPE;
 BEGIN cerinta9('Iasi', output); -- nu exista clienti
 END;
 Error report -
 ORA-20000: Nu exista clienti
 ORA-06512: at "ADMIN.CERINTA9", line 15
 ORA-06512: at line 2
 20000. 00000 - "%s"
 *Cause: The stored procedure 'raise_application_error'
 was called which causes this error to be generated.
 *Action: Correct the problem as described in the error message or contact
 the application administrator or DBA for more information.

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

-- Trigger care sa permita o noua angajare doar în intervalul de ore 8:00 - 16:00, de luni până vineri.

```
CREATE OR REPLACE TRIGGER cerinta10
```

```
BEFORE INSERT ON angajati
```

```
BEGIN
```

```

        IF (TO_CHAR(SYSDATE,'D') = 1) OR TO_CHAR(SYSDATE,'D') = 6 -- duminica e 1, sambata
e 6
            OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 16)
        THEN RAISE_APPLICATION_ERROR(-20001,'In afara programului!');
    END IF;
END;
/

SELECT sysdate FROM dual; -- trebuie sa incalce regula din cerinta
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru)
VALUES (10, 'Toma Cosmin', 1850, 0708227324, 'Ilfov', 5);

```

```

-- 10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.
-- Trigger care sa permita o noua angajare doar în intervalul de ore 8:00 - 16:00, de luni până vineri.
CREATE OR REPLACE TRIGGER cerinta10
BEFORE INSERT ON angajati
BEGIN
    IF (TO_CHAR(SYSDATE,'D') = 1) OR TO_CHAR(SYSDATE,'D') = 6 -- duminica e 1, sambata e 6
        OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 16)
    THEN RAISE_APPLICATION_ERROR(-20001,'In afara programului!');
END IF;
END;
/

SELECT to_char(sysdate,'MON-DD-YY HH24:MI:SS') FROM dual; -- trebuie sa incalce regula din cerinta
INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru)
VALUES (10, 'Toma Cosmin', 1850, 0708227324, 'Ilfov', 5);

rollback;

```

Script Output x Query Result x

Task completed in 0.277 seconds

Error starting at line : 286 in command -

```

INSERT INTO angajati (id_angajat, nume, salariu, telefon, adresa, id_centru)
VALUES (10, 'Toma Cosmin', 1850, 0708227324, 'Ilfov', 5)

```

Error report -

```

ORA-20001: In afara programului!
ORA-06512: at "ADMIN.CERINTA10", line 4
ORA-04088: error during execution of trigger 'ADMIN.CERINTA10'

```

TO_CHAR(SYSDATE,'MON-DD-YYHH24:MI:SS')
1 JAN-06-22 17:07:31

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

-- Trigger care sa nu permita schimbarea specializarii (categoriei de deseuri) a unui client.

```

CREATE OR REPLACE TRIGGER cerinta11
BEFORE UPDATE OF specializare ON clienti
FOR EACH ROW
    WHEN (NEW.specializare <> OLD.specializare)

```

```
BEGIN
```

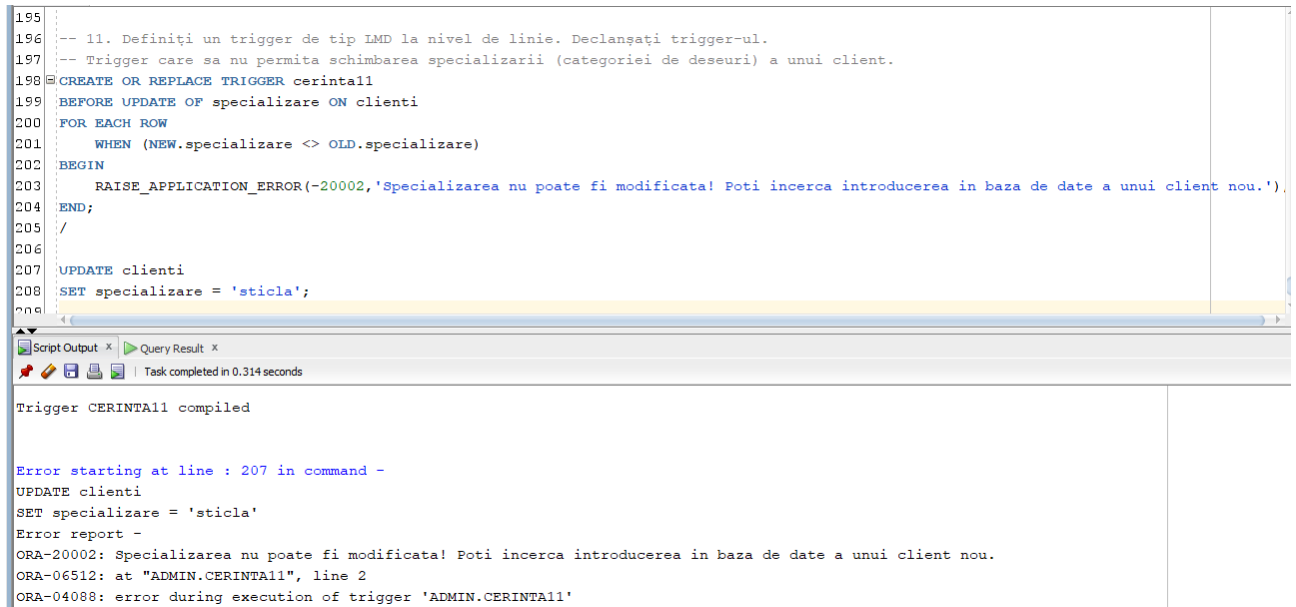
```
    RAISE_APPLICATION_ERROR(-20002,'Specializarea nu poate fi modificata! Poti incerca  
    introducerea in baza de date a unui client nou.');
```

```
END;
```

```
/
```

```
UPDATE clienti
```

```
SET specializare = 'sticla';
```



```
195  
196 -- 11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.  
197 -- Trigger care sa nu permita schimbarea specializării (categoriei de deseuri) a unui client.  
198 CREATE OR REPLACE TRIGGER cerinta11  
199 BEFORE UPDATE OF specializare ON clienti  
200 FOR EACH ROW  
201 WHEN (NEW.specializare <> OLD.specializare)  
202 BEGIN  
203     RAISE_APPLICATION_ERROR(-20002,'Specializarea nu poate fi modificata! Poti incerca introducerea in baza de date a unui client nou.').  
204 END;  
205 /  
206  
207 UPDATE clienti  
208 SET specializare = 'sticla';  
209
```

Script Output x Query Result x

Task completed in 0.314 seconds

Trigger CERINTA11 compiled

Error starting at line : 207 in command -
UPDATE clienti
SET specializare = 'sticla'

Error report -
ORA-20002: Specializarea nu poate fi modificata! Poti incerca introducerea in baza de date a unui client nou.
ORA-06512: at "ADMIN.CERINTA11", line 2
ORA-04088: error during execution of trigger 'ADMIN.CERINTA11'

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

-- Trigger care insereaza intr-un tabel comenzile LDD reusite, iar la erori returneaza mesaj de avertizare.

```
CREATE TABLE tabel_actiuni (utilizator VARCHAR2(30), eveniment VARCHAR2(20), data  
TIMESTAMP(3));
```

```
CREATE OR REPLACE TRIGGER cerinta12
```

```
AFTER CREATE OR DROP OR ALTER OR SERVERERROR ON DATABASE
```

```
BEGIN
```

```
    IF (DBMS_UTILITY.FORMAT_ERROR_STACK IS NULL) THEN
```

```
        INSERT INTO tabel_actiuni VALUES (SYS.LOGIN_USER, SYS.SYSEVENT, SYSTIMESTAMP);
```

```
    ELSE RAISE_APPLICATION_ERROR(-20000, 'Au aparut erori in aplicatie!');
```

```
    END IF;
```

```

END;

/

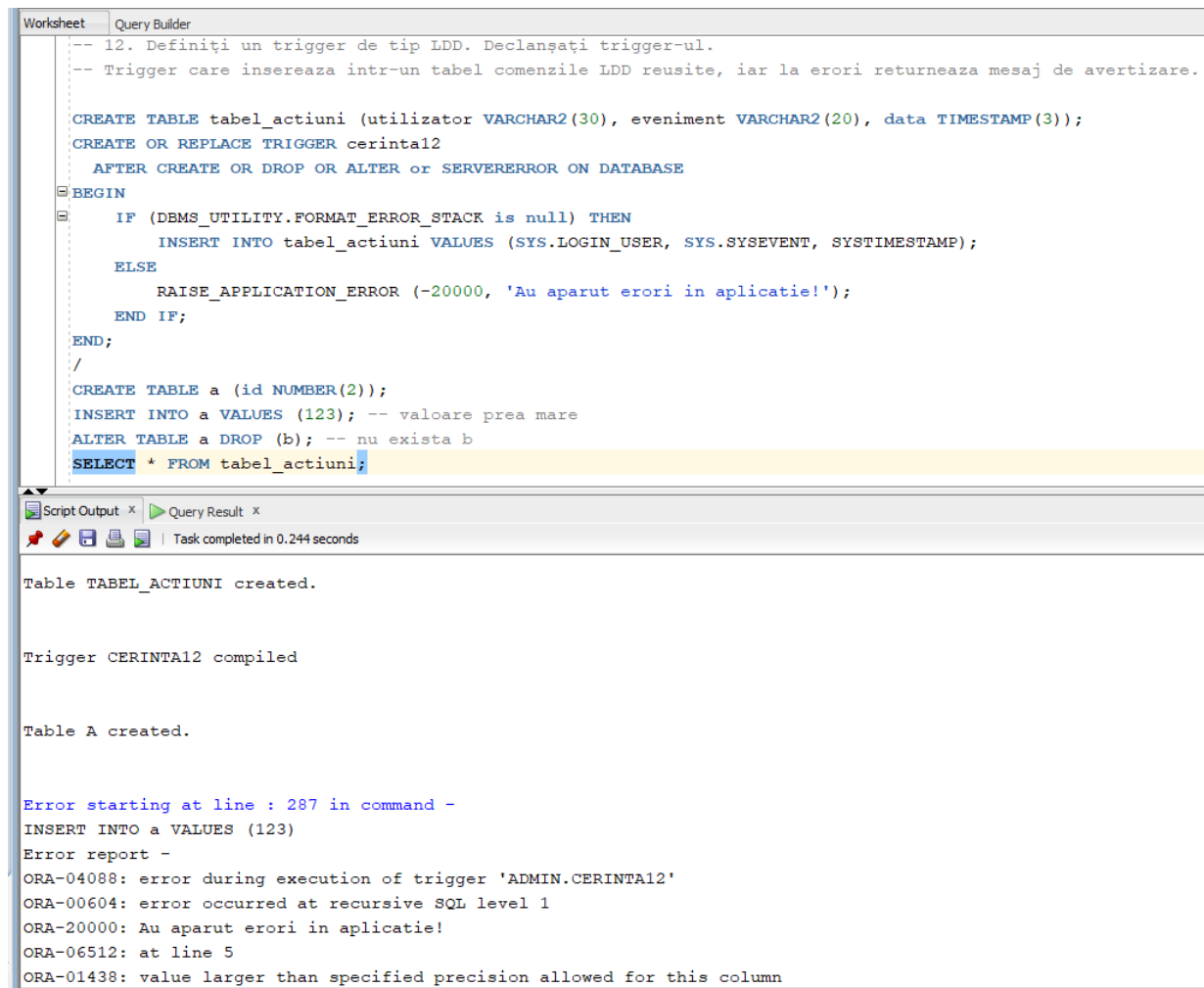
CREATE TABLE a (id NUMBER(2));

INSERT INTO a VALUES (123); -- valoare prea mare

ALTER TABLE a DROP (b); -- nu exista b

SELECT * FROM tabel_actiuni;

```



The screenshot shows the Oracle SQL Developer interface. The top pane is the 'Query Builder' window, which contains a SQL script. The script starts with a comment: '-- 12. Definiți un trigger de tip LDD. Declanșați trigger-ul. -- Trigger care insereaza intr-un tabel comenzile LDD reusite, iar la erori returneaza mesaj de avertizare.' The script then defines a table 'tabel_actiuni' with columns 'utilizator VARCHAR2(30)', 'eveniment VARCHAR2(20)', and 'data TIMESTAMP(3)'. It then creates a trigger 'cerinta12' that fires after create, drop, or alter on the database. The trigger logic is as follows: if 'DBMS_UTILITY.FORMAT_ERROR_STACK' is null, it inserts a record into 'tabel_actiuni' with 'SYS.LOGIN_USER', 'SYS.SYSEVENT', and 'SYSTIMESTAMP'. Otherwise, it raises an application error with message number -20000 and the text 'Au aparut erori in aplicatie!'. The script ends with a semicolon. Below the script, the 'Script Output' pane shows the execution results. It indicates that the table 'TABEL_ACTIUNI' was created, the trigger 'CERINTA12' was compiled, and table 'A' was created. However, an error occurred during the execution of the 'INSERT INTO a VALUES (123)' statement. The error report shows the following messages: 'ORA-04088: error during execution of trigger 'ADMIN.CERINTA12'', 'ORA-00604: error occurred at recursive SQL level 1', 'ORA-20000: Au aparut erori in aplicatie!', 'ORA-06512: at line 5', and 'ORA-01438: value larger than specified precision allowed for this column'.

```

-- 12. Definiți un trigger de tip LDD. Declanșați trigger-ul.
-- Trigger care insereaza intr-un tabel comenzile LDD reusite, iar la erori returneaza mesaj de avertizare.

CREATE TABLE tabel_actiuni (utilizator VARCHAR2(30), eveniment VARCHAR2(20), data TIMESTAMP(3));
CREATE OR REPLACE TRIGGER cerinta12
  AFTER CREATE OR DROP OR ALTER ON DATABASE
BEGIN
  IF (DBMS_UTILITY.FORMAT_ERROR_STACK is null) THEN
    INSERT INTO tabel_actiuni VALUES (SYS.LOGIN_USER, SYS.SYSEVENT, SYSTIMESTAMP);
  ELSE
    RAISE_APPLICATION_ERROR (-20000, 'Au aparut erori in aplicatie!');
  END IF;
END;
/
CREATE TABLE a (id NUMBER(2));
INSERT INTO a VALUES (123); -- valoare prea mare
ALTER TABLE a DROP (b); -- nu exista b
SELECT * FROM tabel_actiuni;

```

Table TABEL_ACTIUNI created.

Trigger CERINTA12 compiled

Table A created.

Error starting at line : 287 in command -
 INSERT INTO a VALUES (123)
 Error report -
 ORA-04088: error during execution of trigger 'ADMIN.CERINTA12'
 ORA-00604: error occurred at recursive SQL level 1
 ORA-20000: Au aparut erori in aplicatie!
 ORA-06512: at line 5
 ORA-01438: value larger than specified precision allowed for this column


```
ALTER TABLE a DROP (b); -- nu exista b
SELECT * FROM tabel_actiuni;
```

Script Output x Query Result x

Task completed in 0.222 seconds

Error starting at line : 285 in command -
ALTER TABLE a DROP (b)
Error report -
ORA-04088: error during execution of trigger 'ADMIN.CERINTA12'
ORA-00604: error occurred at recursive SQL level 1
ORA-20000: Au aparut erori in aplicatie!
ORA-06512: at line 5
ORA-00904: "B": invalid identifier
04088. 00000 - "error during execution of trigger '%s.%s'"
*Cause: A runtime error occurred during execution of a trigger.
*Action: Check the triggers which were involved in the operation.

```
SELECT * FROM tabel_actiuni;
```

Script Output x Query Result x

All Rows Fetched: 1 in 0.048 seconds

	UTILIZATOR	EVENTIMENT	DATA
1	ADMIN	CREATE	04-JAN-22 09.45.57.946000000 AM

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

```
CREATE OR REPLACE PACKAGE cerinta13 AS
    PROCEDURE cerinta6 (loc centre_de_colectare.locatie%TYPE);
    PROCEDURE cerinta7;
    FUNCTION cerinta8 (centrulDat centre_de_colectare.id_centru%type) RETURN VARCHAR2;
    PROCEDURE cerinta9 (v_judet IN locatii.judet%TYPE, raspuns OUT clienti%ROWTYPE);
END cerinta13;
/

CREATE OR REPLACE PACKAGE BODY cerinta13 AS
    PROCEDURE cerinta6 (loc centre_de_colectare.locatie%TYPE) IS
        TYPE tablou IS TABLE OF clienti%ROWTYPE INDEX BY PLS_INTEGER;
        t tablou;
        type vector is varray(20) of number(3); -- contine id-urile cosurilor de gunoi
        v vector := vector();
    BEGIN
        SELECT id_cos BULK COLLECT INTO v
        FROM cosuri_de_gunoi
```

```

        WHERE oras = loc;

        DBMS_OUTPUT.PUT_LINE('Exista ' || v.COUNT || ' cosuri de gunoi pentru locatia ' || loc);

```

```

        SELECT * BULK COLLECT INTO t FROM clienti WHERE adresa = loc;

```

```

        DBMS_OUTPUT.PUT_LINE('Exista ' || t.COUNT || ' clienti pentru centrul de colectare din ' || loc);

```

```

        FOR i IN t.FIRST..t.LAST LOOP DBMS_OUTPUT.PUT_LINE(' -> Clientul ' || t(i).id_client || ': ' || t(i).nume);

```

```

        END LOOP;

```

```

        FOR j IN v.FIRST..v.LAST LOOP DBMS_OUTPUT.PUT_LINE(' -> Cosul de gunoi ' || v(j));

```

```

        END LOOP;

```

```

    END;

```

```

PROCEDURE cerinta7 IS

```

```

    v_centru centre_de_colectare.id_centru%TYPE;

```

```

    v_nr NUMBER(4);

```

```

    CURSOR c IS SELECT c.id_centru, COUNT(a.id_angajat) nrAng
                FROM angajati a RIGHT JOIN centre_de_colectare c ON (c.id_centru = a.id_centru)

```

```

                GROUP BY c.id_centru ORDER BY c.id_centru;

```

```

    BEGIN

```

```

        OPEN c;

```

```

        LOOP

```

```

            FETCH c INTO v_centru, v_nr;

```

```

            EXIT WHEN c%NOTFOUND;

```

```

            IF v_nr = 0 THEN DBMS_OUTPUT.PUT_LINE('La centrul ' || v_centru || ' nu lucreaza angajati.');
```

```

            ELSIF v_nr = 1 THEN DBMS_OUTPUT.PUT_LINE('La centrul ' || v_centru || ' lucreaza un angajat.');
```

```

            ELSE DBMS_OUTPUT.PUT_LINE('La centrul ' || v_centru || ' lucreaza ' || v_nr || ' angajati.');
```

```

            END IF;

```

```

        END LOOP;

```

```

        CLOSE c;

```

```

    END;

```

```

FUNCTION cerinta8(centrulDat centre_de_colectare.id_centru%TYPE) RETURN VARCHAR2 IS
    TYPE rec IS RECORD (nrAng NUMBER, v_ume VARCHAR2(30), v_adresa VARCHAR2(30));
    ang rec;
    TYPE vector IS VARRAY(20) OF centre_de_colectare.id_centru%TYPE;
    v vector; -- neinitializat, adica fara :=vector();
BEGIN
    -- am vrut sa evidentiez si folosirea exceptiei collection_is_null, asa ca am
    -- facut un if special care sa o apeleze pentru inputul 0
    IF centrulDat = 0 THEN v(1) := centrulDat; END IF;

    SELECT COUNT(id_angajat), nume, adresa INTO ang
    FROM angajati a                                -- primul tabel
    WHERE id_centru = centrulDat AND id_centru in
        (SELECT id_centru FROM soferi                -- al doilea
         WHERE nume = a.nume and adresa in (SELECT locatie FROM
centre_de_colectare))                               -- al treilea
    GROUP BY nume, adresa ORDER BY id_centru;

    RETURN ('Pentru centrul ' || centrulDat || ' avem ' || ang.nrAng || ' angajati
soferi.');
```

```

EXCEPTION
    WHEN no_data_found THEN RETURN 'Nu exista.';
    WHEN collection_is_null THEN RETURN 'Colectie nula';
    WHEN others THEN RETURN 'Alta exceptie!';
END;
```

```

PROCEDURE cerinta9 (v_judet IN locatii.judet%TYPE, raspuns OUT clienti%ROWTYPE) IS
BEGIN
    select * into raspuns from clienti where id_client in (
        select id_client from client_categorie where id_categorie in (
            select id_categorie from centru_categorie where id_centru in (
                select id_centru from cosuri_de_gunoi where id_cos in (
                    select id_cos from locatii where lower(judet) =
lower(v_judet))));
    EXCEPTION
```

```

        WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20000, 'Nu exista clienti');
        WHEN TOO_MANY_ROWS THEN RAISE_APPLICATION_ERROR(-20001, 'Exista mai multi
        clienti');
        WHEN OTHERS THEN RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
    END;
END;
/

BEGIN cerinta13.cerinta6('Bucuresti'); END;
/
BEGIN cerinta13.cerinta7(); END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('-> Pentru id_centru inexistent:');
    DBMS_OUTPUT.PUT_LINE(cerinta13.cerinta8(70)); -- centru fara angajati / neinreg in
    BD
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('-> Pentru collection_is_null:');
    DBMS_OUTPUT.PUT_LINE(cerinta13.cerinta8(0)); -- if special pt 0
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('-> Pentru centru cu angajati soferi (deci nicio eroare
    aruncata):');
    DBMS_OUTPUT.PUT_LINE(cerinta13.cerinta8(2)); -- merge cu codurile 1-5
END;
/
BEGIN
    DBMS_OUTPUT.PUT_LINE('-> Pentru centru fara angajati:');
    DBMS_OUTPUT.PUT_LINE(cerinta13.cerinta8(6));
END;
/
DECLARE output clienti%ROWTYPE;
BEGIN cerinta13.cerinta9('Arad', output);
    DBMS_OUTPUT.PUT_LINE('Cerinta9: clientul cu id-ul ' || output.id_client || ' si
    numele ' || output.numa);

```

```

END;

/

DECLARE output clienti%ROWTYPE;

BEGIN cerinta13.cerinta9('Prahova', output); -- exista mai multi clienti

END;

/

DECLARE output clienti%ROWTYPE;

BEGIN cerinta13.cerinta9('Iasi', output); -- nu exista clienti

END;

/

```

```

330 -- 13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.
331 CREATE OR REPLACE PACKAGE cerinta13 AS
332     PROCEDURE cerinta6 (loc centre_de_colectare.locatie%TYPE);
333     PROCEDURE cerinta7;
334     FUNCTION cerinta8 (centrulDat centre_de_colectare.id_centru%TYPE) RETURN VARCHAR2;
335     PROCEDURE cerinta9 (v_judet IN locatii.judet%TYPE, raspuns OUT clienti%ROWTYPE);
336 END cerinta13;
337 /
338 CREATE OR REPLACE PACKAGE BODY cerinta13 AS
339     PROCEDURE cerinta6 (loc centre_de_colectare.locatie%TYPE) IS
340         TYPE tablou IS TABLE OF clienti%ROWTYPE INDEX BY PLS_INTEGER;
341         t tablou;
342         type vector is varray(20) of number(3); -- va contine id-urile cosurilor de gunoi
343         v vector := vector();
344     BEGIN
345         SELECT id_cos BULK COLLECT INTO v
346         FROM cosuri_de_gunoi
347         WHERE oras = loc;
348         DBMS_OUTPUT.PUT_LINE('Exista ' || v.COUNT || ' cosuri de gunoi pentru locatia ' || loc);
349
350         SELECT * BULK COLLECT INTO t
351         FROM clienti
352         WHERE adresa = loc;
353
354         DBMS_OUTPUT.PUT_LINE('Exista ' || t.COUNT || ' clienti pentru centrul de colectare din ' || loc);

```

Script Output x Query Result x

Task completed in 0.735 seconds

Package CERINTA13 compiled

Package Body CERINTA13 compiled

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).

```

-- Pentru fiecare utilizator, afisez id-urile cosurilor de gunoi folosite.

CREATE OR REPLACE PACKAGE pachet14 IS

```

```

PROCEDURE gaseste_utilizatori;
FUNCTION obtine_adresa (i NUMBER) RETURN utilizatori.adresa%TYPE;
FUNCTION obtine_concat (i NUMBER) RETURN VARCHAR2;
PROCEDURE afis;
END;
/

CREATE OR REPLACE PACKAGE BODY pachet14 AS
    TYPE tablou IS TABLE OF VARCHAR2(100) INDEX BY PLS_INTEGER;
    t tablou;
    TYPE vector IS VARRAY(20) OF NUMBER(3); -- va contine id-urile cosurilor de gunoi
    v vector := vector();

    PROCEDURE gaseste_utilizatori IS
    BEGIN
        SELECT distinct(u.id_utilizator) bulk collect INTO t
        FROM utilizatori u JOIN utilizator_cos uc ON u.id_utilizator=uc.id_utilizator
        JOIN cosuri_de_gunoi cdg ON uc.id_cos = cdg.id_cos
        WHERE to_char(data_instalarii, 'YY') >= '15';
    END gaseste_utilizatori;

    FUNCTION obtine_adresa (i NUMBER) RETURN utilizatori.adresa%TYPE IS rez
    utilizatori.adresa%TYPE;
    BEGIN
        SELECT adresa INTO rez FROM utilizatori WHERE id_utilizator = i;
        RETURN rez;
    END;

    FUNCTION obtine_concat (i NUMBER) RETURN VARCHAR2 IS rez VARCHAR2(100);
    BEGIN rez := 'Cosul '|| i; RETURN rez;
    END;

    PROCEDURE afis IS
    BEGIN
        gaseste_utilizatori(); -- am populat tabelul indexat
        FOR i IN t.first..t.last LOOP
            DBMS_OUTPUT.PUT_LINE('-----');
        END LOOP;
    END;
END pachet14;

```

```

        SELECT id_cos BULK COLLECT INTO v FROM utilizator_cos WHERE id_utilizator
        = t(i); -- am populat vectorul

        DBMS_OUTPUT.PUT_LINE('Utilizatorul cu id-ul ' || i || ' de la adresa ' ||
        obtine_adresa(i) || ' a folosit ');

        FOR j IN v.first..v.last LOOP -- pentru fiecare cos folosit

            DBMS_OUTPUT.PUT_LINE(obtine_concat(v(j)));

        END LOOP;

    END LOOP;

END afis;

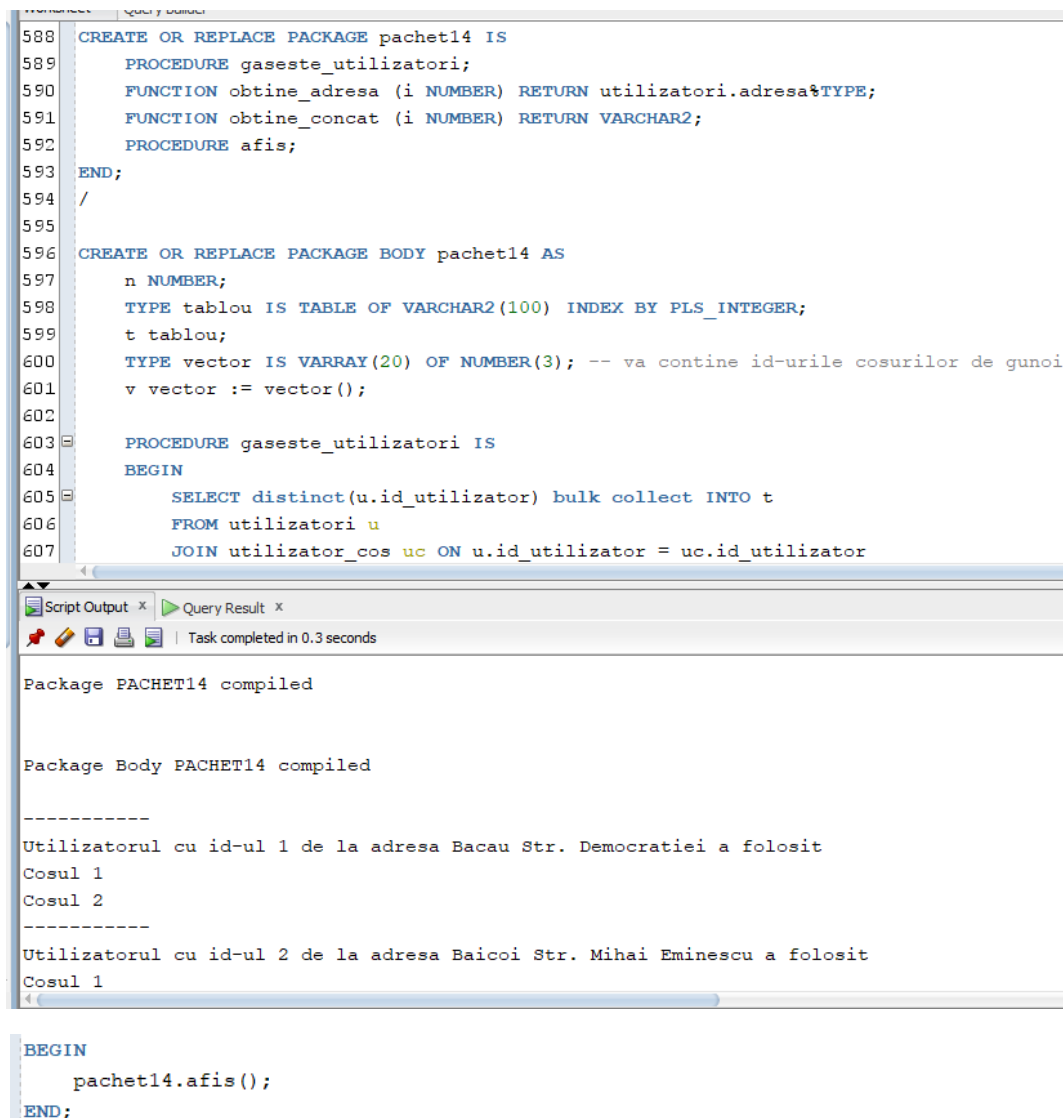
END pachet14;

/

BEGIN pachet14.afis();

END;

```



```

588 CREATE OR REPLACE PACKAGE pachet14 IS
589     PROCEDURE gaseste_utilizatori;
590     FUNCTION obtine_adresa (i NUMBER) RETURN utilizatori.adresa%TYPE;
591     FUNCTION obtine_concat (i NUMBER) RETURN VARCHAR2;
592     PROCEDURE afis;
593 END;
594 /
595
596 CREATE OR REPLACE PACKAGE BODY pachet14 AS
597     n NUMBER;
598     TYPE tablou IS TABLE OF VARCHAR2(100) INDEX BY PLS_INTEGER;
599     t tablou;
600     TYPE vector IS VARRAY(20) OF NUMBER(3); -- va contine id-urile cosurilor de gunoi
601     v vector := vector();
602
603     PROCEDURE gaseste_utilizatori IS
604     BEGIN
605         SELECT distinct(u.id_utilizator) bulk collect INTO t
606         FROM utilizatori u
607         JOIN utilizator_cos uc ON u.id_utilizator = uc.id_utilizator

```

Script Output x Query Result x

Task completed in 0.3 seconds

```

Package PACHET14 compiled

Package Body PACHET14 compiled

-----
Utilizatorul cu id-ul 1 de la adresa Bacau Str. Democratiei a folosit
Cosul 1
Cosul 2
-----
Utilizatorul cu id-ul 2 de la adresa Baiccoi Str. Mihai Eminescu a folosit
Cosul 1

```

```

BEGIN
    pachet14.afis();
END;

```