

Gestionarea unui parc auto

Maftai Ștefan

Seria 3

Grupa 342

An universitar 2024-2025

Cuprins

1. Prezențați pe scurt baza de date (utilitatea ei).....	5
2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD, model de diagrama entitate-relație; nu se va accepta alt format).....	6
3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.....	7
4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc).....	8
5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).....	11
6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.....	17
7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.....	20
8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.....	22
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.....	26
10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.....	30

11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.....	32
12. Definiți un trigger de tip LDD. Declanșați trigger-ul.....	33

Introducere:

1. Tema proiectului este gestiunea unui parc de vehicule.
2. Versiunea SGBD-ului: SQLDeveloper 21.4.4, Oracle Database 21c.
3. Configurație Hardware: procesor Intel Core i5 9300HF, placa video Nvidia GeForce GTX 1650, 16 GB RAM DDR4, SSD 128GB, HDD 1TB.
4. Alcoare memoria RAM: ~ 800MB de RAM pentru SQLDeveloper am observat în Task Manager, iar pentru SQL Plus ~ 200-400 MB

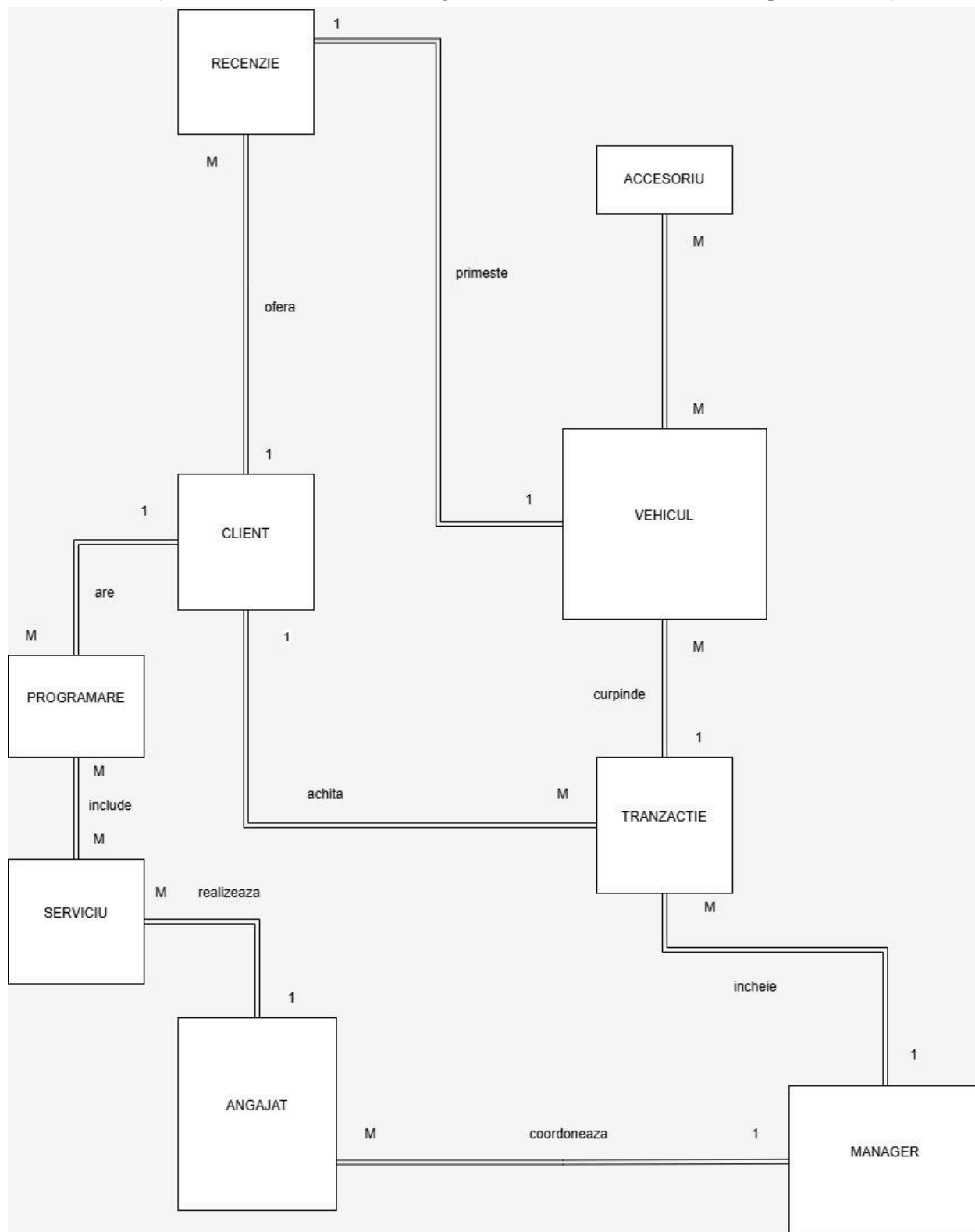
>	 sqldeveloper64W	1.3%	854.2 MB	0 MB/s	0 Mbps
>	 Oracle RDBMS Kernel Executa...	0%	253.3 MB	0.1 MB/s	0 Mbps

- 5.
6. Sistem de Operare: Windows 11 Pro.
7. Nu s-a utilizat mașina virtuală.

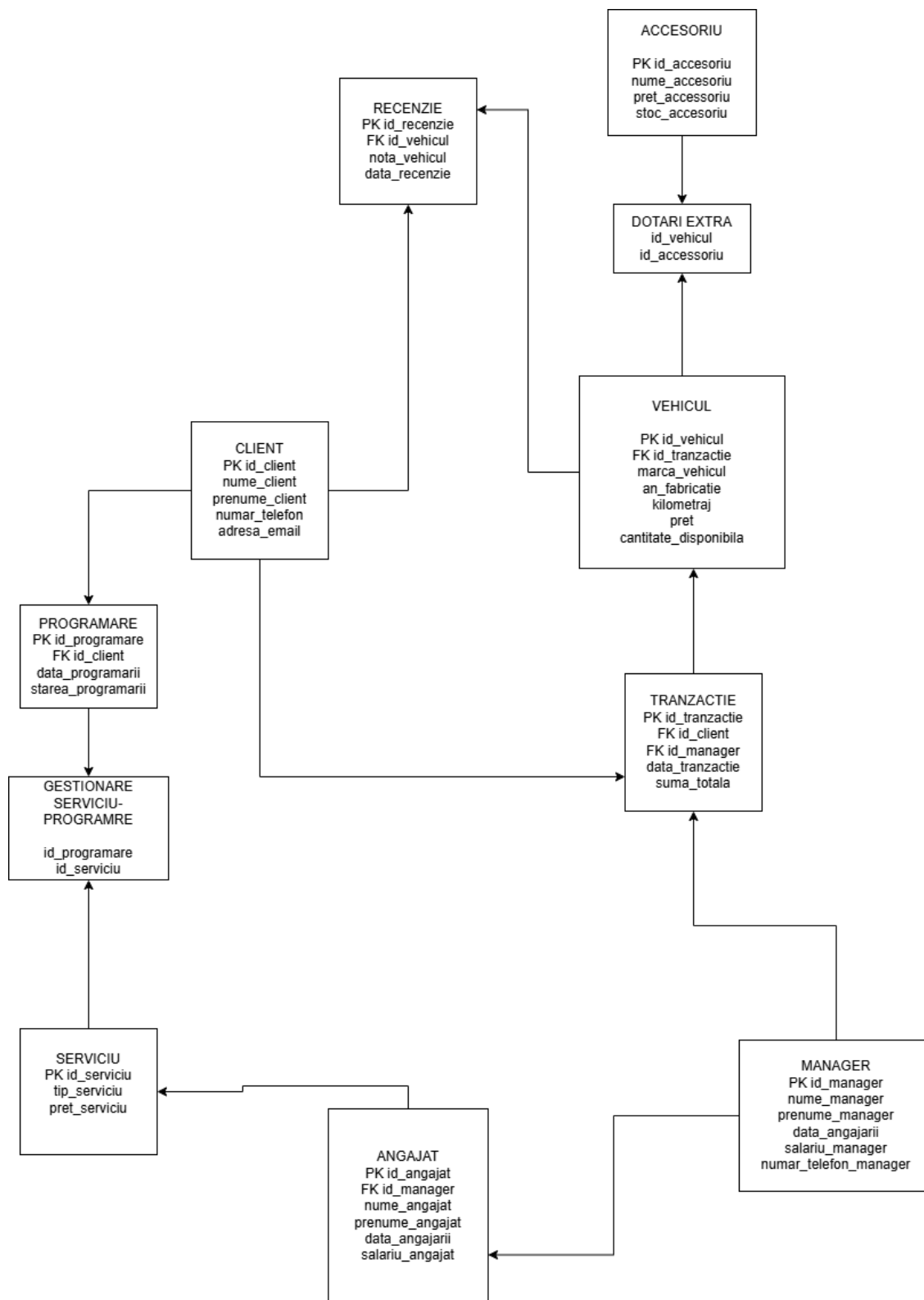
1. Prezența pe scurt baza de date (utilitatea ei)

Baza de date este concepută pentru administrarea unui parc de vehicule, aceasta centralizează toate informațiile la un loc. Aceasta conține clienți care își pot face programări pentru servicii auto, să ofere recenzii pentru mașinile din parc și să facă diferite tranzacții. Aceste acțiuni sunt realizate de angajați, care la rândul lor sunt gestionate de manageri.

2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD, model de diagrama entitate-relație; nu se va accepta alt format)



3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.



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

- Creerea tabelor:

```
CREATE TABLE CLIENT (  
    id_client NUMBER PRIMARY KEY,  
    nume_client VARCHAR2(50) NOT NULL,  
    prenume_client VARCHAR2(50) NOT NULL,  
    numar_telefon VARCHAR2(15) NOT NULL,  
    adresa_email VARCHAR2(100) NOT NULL  
);
```

```
CREATE TABLE PROGRAMARE (  
    id_programare NUMBER PRIMARY KEY,  
    id_client NUMBER NOT NULL,  
    data_programarii DATE NOT NULL,  
    starea_programarii VARCHAR2(50) NOT NULL,  
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client)  
);
```

```
CREATE TABLE SERVICIU (  
    id_serviciu NUMBER PRIMARY KEY,  
    tip_serviciu VARCHAR2(50) NOT NULL,  
    pret_serviciu NUMBER(10,2) NOT NULL  
);
```

```
CREATE TABLE GESTIONARE_SERVICIU_PROGRAMARE (  
    id_programare NUMBER NOT NULL,  
    id_serviciu NUMBER NOT NULL,  
    PRIMARY KEY (id_programare, id_serviciu),  
    FOREIGN KEY (id_programare) REFERENCES PROGRAMARE(id_programare),  
    FOREIGN KEY (id_serviciu) REFERENCES SERVICIU(id_serviciu)  
);
```

```
CREATE TABLE MANAGER (  
    id_manager NUMBER PRIMARY KEY,  
    nume_manager VARCHAR2(50) NOT NULL,  
    prenume_manager VARCHAR2(50) NOT NULL,  
    data_angajarii DATE NOT NULL,  
    salariu_manager NUMBER(10,2) NOT NULL,  
    numar_telefon_manager VARCHAR2(15) NOT NULL  
);
```

```
CREATE TABLE ANGAJAT (  
    id_angajat NUMBER PRIMARY KEY,  
    id_manager NUMBER NOT NULL,  
    nume_angajat VARCHAR2(50) NOT NULL,  
    prenume_angajat VARCHAR2(50) NOT NULL,  
    data_angajarii DATE NOT NULL,
```



```
    salariu_angajat NUMBER(10,2) NOT NULL,  
    FOREIGN KEY (id_manager) REFERENCES MANAGER(id_manager)  
);  
  
CREATE TABLE TRANZACTIE (  
    id_tranzactie NUMBER PRIMARY KEY,  
    id_client NUMBER NOT NULL,  
    id_manager NUMBER NOT NULL,  
    data_tranzactie DATE NOT NULL,  
    suma_totala NUMBER(10,2) NOT NULL,  
    FOREIGN KEY (id_client) REFERENCES CLIENT(id_client),  
    FOREIGN KEY (id_manager) REFERENCES MANAGER(id_manager)  
);  
  
CREATE TABLE VEHICUL (  
    id_vehicul NUMBER PRIMARY KEY,  
    id_tranzactie NUMBER NOT NULL,  
    marca_vehicul VARCHAR2(50) NOT NULL,  
    an_fabricatie NUMBER NOT NULL,  
    kilometraj NUMBER NOT NULL,  
    pret NUMBER(10,2) NOT NULL,  
    cantitate_disponibila NUMBER NOT NULL,  
    FOREIGN KEY (id_tranzactie) REFERENCES TRANZACTIE(id_tranzactie)  
);  
  
CREATE TABLE RECENZIE (  
    id_recenzie NUMBER PRIMARY KEY,  
    id_vehicul NUMBER NOT NULL,  
    nota_vehicul NUMBER(2) NOT NULL CHECK (nota_vehicul BETWEEN 1 AND 10),  
    data_recenzie DATE NOT NULL,  
    FOREIGN KEY (id_vehicul) REFERENCES VEHICUL(id_vehicul) ON DELETE CASCADE  
);  
  
CREATE TABLE ACCESORIU (  
    id_accesoriu NUMBER PRIMARY KEY,  
    nume_accesoriu VARCHAR2(50) NOT NULL,  
    pret_accesoriu NUMBER(10,2) NOT NULL,  
    stoc_accesoriu NUMBER NOT NULL  
);  
  
CREATE TABLE DOTARI_EXTRA (  
    id_vehicul NUMBER NOT NULL,  
    id_accesoriu NUMBER NOT NULL,  
    PRIMARY KEY (id_vehicul, id_accesoriu),  
    FOREIGN KEY (id_vehicul) REFERENCES VEHICUL(id_vehicul),  
    FOREIGN KEY (id_accesoriu) REFERENCES ACCESORIU(id_accesoriu)  
);
```

- Constrângeri întâlnite:
 - NOT NULL - este folosită ca unele coloane să nu aibe valori NULL. Acest lucru garantează că înregistrările din tabel vor conține date.
 - Primary Key - este folosită pentru unicitatea valorilor din coloanele respective și pentru a avea fiecare tabel o identificare unică.

- Foreign Key - definirea unei relații dintre doua tabele.
- ON DELETE CASCADE - se folosește pentru că atunci când se șterge înregistrarea unui rând din tabelul principal, atunci toate înregistrările se vor șterge automat din cel secundar.
- CHECK - condiție suplimentară asupra valorilor unei coloane. De exemplu, pentru *nota_vehicul* valorile sunt cuprins între 1 și 10.

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

```
CREATE SEQUENCE client_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE serviciu_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE manager_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE accesoriu_seq START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE programare_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE angajat_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE tranzactie_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE vehicul_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE recenzie_seq START WITH 1 INCREMENT BY 1;
```

```
INSERT INTO CLIENT VALUES (client_seq.NEXTVAL, 'Mihai', 'George', '0721123456', 'mihai.george@gmail.com');
INSERT INTO CLIENT VALUES (client_seq.NEXTVAL, 'Maftei', 'Stefan', '0732123456', 'maftei.stefan@gmail.com');
INSERT INTO CLIENT VALUES (client_seq.NEXTVAL, 'Vasile', 'Ana', '0743123456', 'vasile.ana@gmail.com');
INSERT INTO CLIENT VALUES (client_seq.NEXTVAL, 'Botezatu', 'Cosmin', '0754123456', 'botezatu.cosmin@gmail.com');
INSERT INTO CLIENT VALUES (client_seq.NEXTVAL, 'Ursu', 'Vlad', '0765123456', 'ursu.vlad@gmail.com');
```

SELECT * FROM CLIENT;

ID_CLIENT	NUME_CLIENT	PRENUME_CLIENT	NUMAR_TELEFON	ADRESA_EMAIL
1	Mihai	George	0721123456	mihai.george@gmail.com
2	Maftei	Stefan	0732123456	maftei.stefan@gmail.com
3	Vasile	Ana	0743123456	vasile.ana@gmail.com
4	Botezatu	Cosmin	0754123456	botezatu.cosmin@gmail.com
5	Ursu	Vlad	0765123456	ursu.vlad@gmail.com

```
INSERT INTO SERVICIU VALUES (serviciu_seq.NEXTVAL, 'Schimb ulei', 150.00);
INSERT INTO SERVICIU VALUES (serviciu_seq.NEXTVAL, 'Revizie generala', 500.00);
INSERT INTO SERVICIU VALUES (serviciu_seq.NEXTVAL, 'Rotire anvelope', 100.00);
INSERT INTO SERVICIU VALUES (serviciu_seq.NEXTVAL, 'Diagnosticare', 200.00);
INSERT INTO SERVICIU VALUES (serviciu_seq.NEXTVAL, 'Spalare exterior', 50.00);
```

SELECT * FROM SERVICIU;

ID_SERVICIU	TIP_SERVICIU	PRET_SERVICIU
1	Schimb ulei	150
2	Revizie generala	500
3	Rotire anvelope	100
4	Diagnosticare	200
5	Spalare exterior	50

```
INSERT INTO MANAGER VALUES (manager_seq.NEXTVAL, 'Chitu', 'Tudor', TO_DATE('2020-01-15', 'YYYY-MM-DD'), 7500.00, '0722123456');
INSERT INTO MANAGER VALUES (manager_seq.NEXTVAL, 'Serban', 'Andrei', TO_DATE('2019-03-20', 'YYYY-MM-DD'), 8000.00, '0733123456');
INSERT INTO MANAGER VALUES (manager_seq.NEXTVAL, 'Enescu', 'Cristina', TO_DATE('2021-05-10', 'YYYY-MM-DD'), 7000.00, '0744123456');
INSERT INTO MANAGER VALUES (manager_seq.NEXTVAL, 'Neagu', 'Emilia', TO_DATE('2018-07-01', 'YYYY-MM-DD'), 8500.00, '0755123456');
```

```
INSERT INTO MANAGER VALUES (manager_seq.NEXTVAL, 'Marin', 'Marian', TO_DATE('2022-02-14', 'YYYY-MM-DD'),
7200.00, '0766123456');
```

```
SELECT * FROM MANAGER;
```



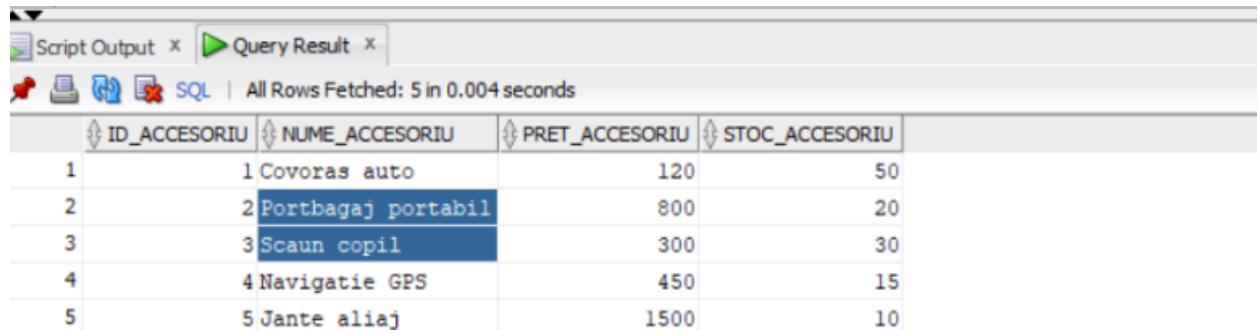
Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.004 seconds

ID_MANAGER	NUME_MANAGER	PRENUME_MANAGER	DATA_ANGAJARII	SALARIU_MANAGER	NUMAR_TELEFON_MANAGER
1	1 Chitu	Tudor	15-JAN-20	7500	0722123456
2	2 Serban	Andrei	20-MAR-19	8000	0733123456
3	3 Enescu	Cristina	10-MAY-21	7000	0744123456
4	4 Neagu	Emilia	01-JUL-18	8500	0755123456
5	5 Marin	Marian	14-FEB-22	7200	0766123456

```
INSERT INTO ACCESORIU VALUES (accesoriu_seq.NEXTVAL, 'Covoras auto', 120.00, 50);
INSERT INTO ACCESORIU VALUES (accesoriu_seq.NEXTVAL, 'Portbagaj portabil', 800.00, 20);
INSERT INTO ACCESORIU VALUES (accesoriu_seq.NEXTVAL, 'Scaun copil', 300.00, 30);
INSERT INTO ACCESORIU VALUES (accesoriu_seq.NEXTVAL, 'Navigatie GPS', 450.00, 15);
INSERT INTO ACCESORIU VALUES (accesoriu_seq.NEXTVAL, 'Jante aliaj', 1500.00, 10);
```

```
SELECT * FROM ACCESORIU;
```




Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.004 seconds

ID_ACCESORIU	NUME_ACCESORIU	PRET_ACCESORIU	STOC_ACCESORIU
1	1 Covoras auto	120	50
2	2 Portbagaj portabil	800	20
3	3 Scaun copil	300	30
4	4 Navigatie GPS	450	15
5	5 Jante aliaj	1500	10

```
INSERT INTO PROGRAMARE VALUES (programare_seq.NEXTVAL, 1, TO_DATE('2023-10-10', 'YYYY-MM-DD'),
'Confirmata');
INSERT INTO PROGRAMARE VALUES (programare_seq.NEXTVAL, 2, TO_DATE('2023-10-11', 'YYYY-MM-DD'), 'In
asteptare');
INSERT INTO PROGRAMARE VALUES (programare_seq.NEXTVAL, 3, TO_DATE('2023-10-12', 'YYYY-MM-DD'),
'Anulata');
INSERT INTO PROGRAMARE VALUES (programare_seq.NEXTVAL, 4, TO_DATE('2023-10-13', 'YYYY-MM-DD'),
'Confirmata');
INSERT INTO PROGRAMARE VALUES (programare_seq.NEXTVAL, 5, TO_DATE('2023-10-14', 'YYYY-MM-DD'),
'Confirmata');
```

```
SELECT * FROM PROGRAMARE;
```



Script Output x Query Result x

All Rows Fetched: 5 in 0.006 seconds

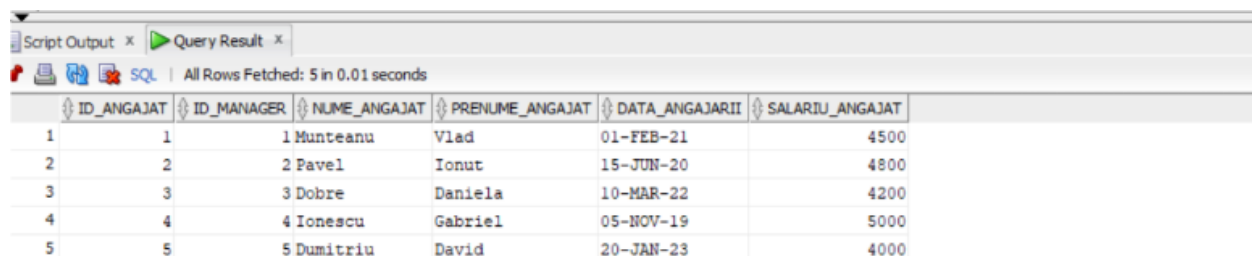
ID_PROGRAMARE	ID_CLIENT	DATA_PROGRAMARII	STAREA_PROGRAMARII
1	1	10-OCT-23	Confirmata
2	2	11-OCT-23	In asteptare
3	3	12-OCT-23	Anulata
4	4	13-OCT-23	Confirmata
5	5	14-OCT-23	Confirmata

```

INSERT INTO ANGAJAT VALUES (angajat_seq.NEXTVAL, 1, 'Munteanu', 'Vlad', TO_DATE('2021-02-01',
'YYYY-MM-DD'), 4500.00);
INSERT INTO ANGAJAT VALUES (angajat_seq.NEXTVAL, 2, 'Pavel', 'Ionut', TO_DATE('2020-06-15', 'YYYY-MM-DD'),
4800.00);
INSERT INTO ANGAJAT VALUES (angajat_seq.NEXTVAL, 3, 'Dobre', 'Daniela', TO_DATE('2022-03-10', 'YYYY-MM-DD'),
4200.00);
INSERT INTO ANGAJAT VALUES (angajat_seq.NEXTVAL, 4, 'Ionescu', 'Gabriel', TO_DATE('2019-11-05',
'YYYY-MM-DD'), 5000.00);
INSERT INTO ANGAJAT VALUES (angajat_seq.NEXTVAL, 5, 'Dumitriu', 'David', TO_DATE('2023-01-20',
'YYYY-MM-DD'), 4000.00);

```

SELECT * FROM ANGAJAT;



Script Output x Query Result x

All Rows Fetched: 5 in 0.01 seconds

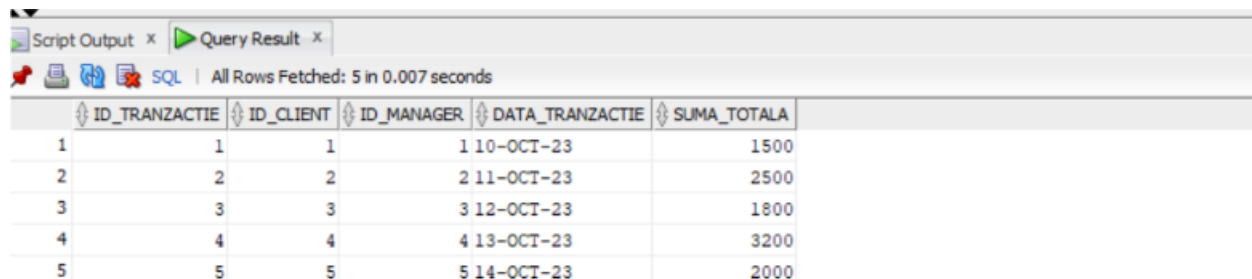
ID_ANGAJAT	ID_MANAGER	NUME_ANGAJAT	PRENUME_ANGAJAT	DATA_ANGAJARII	SALARIU_ANGAJAT
1	1	1 Munteanu	Vlad	01-FEB-21	4500
2	2	2 Pavel	Ionut	15-JUN-20	4800
3	3	3 Dobre	Daniela	10-MAR-22	4200
4	4	4 Ionescu	Gabriel	05-NOV-19	5000
5	5	5 Dumitriu	David	20-JAN-23	4000

```

INSERT INTO TRANZACTIE VALUES (tranzactie_seq.NEXTVAL, 1, 1, TO_DATE('2023-10-10', 'YYYY-MM-DD'),
1500.00);
INSERT INTO TRANZACTIE VALUES (tranzactie_seq.NEXTVAL, 2, 2, TO_DATE('2023-10-11', 'YYYY-MM-DD'),
2500.00);
INSERT INTO TRANZACTIE VALUES (tranzactie_seq.NEXTVAL, 3, 3, TO_DATE('2023-10-12', 'YYYY-MM-DD'),
1800.00);
INSERT INTO TRANZACTIE VALUES (tranzactie_seq.NEXTVAL, 4, 4, TO_DATE('2023-10-13', 'YYYY-MM-DD'),
3200.00);
INSERT INTO TRANZACTIE VALUES (tranzactie_seq.NEXTVAL, 5, 5, TO_DATE('2023-10-14', 'YYYY-MM-DD'),
2000.00);

```

SELECT * FROM TRANZACTIE;



Script Output x Query Result x

All Rows Fetched: 5 in 0.007 seconds

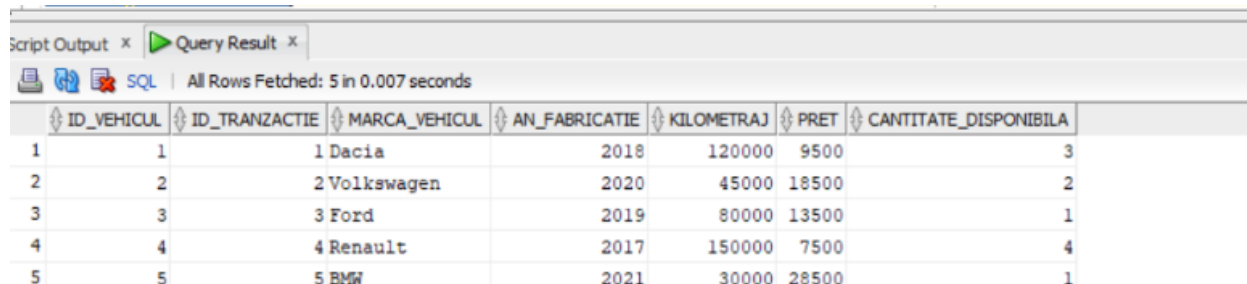
ID_TRANZACTIE	ID_CLIENT	ID_MANAGER	DATA_TRANZACTIE	SUMA_TOTALA
1	1	1	10-OCT-23	1500
2	2	2	11-OCT-23	2500
3	3	3	12-OCT-23	1800
4	4	4	13-OCT-23	3200
5	5	5	14-OCT-23	2000

Maftei Ștefan, grupa 342

Proiect SGBD

```
INSERT INTO VEHICUL VALUES (vehicul_seq.NEXTVAL, 1, 'Dacia', 2018, 120000, 9500.00, 3);
INSERT INTO VEHICUL VALUES (vehicul_seq.NEXTVAL, 2, 'Volkswagen', 2020, 45000, 18500.00, 2);
INSERT INTO VEHICUL VALUES (vehicul_seq.NEXTVAL, 3, 'Ford', 2019, 80000, 13500.00, 1);
INSERT INTO VEHICUL VALUES (vehicul_seq.NEXTVAL, 4, 'Renault', 2017, 150000, 7500.00, 4);
INSERT INTO VEHICUL VALUES (vehicul_seq.NEXTVAL, 5, 'BMW', 2021, 30000, 28500.00, 1);
```

SELECT * FROM VEHICUL;



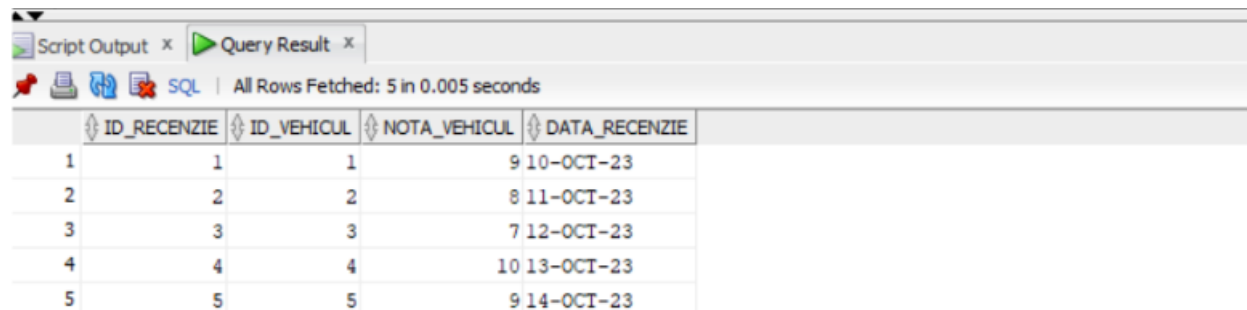
Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.007 seconds

ID_VEHICUL	ID_TRANZACTIE	MARCA_VEHICUL	AN_FABRICATIE	KILOMETRAJ	PRET	CANTITATE_DISPONIBILA
1	1	1 Dacia	2018	120000	9500	3
2	2	2 Volkswagen	2020	45000	18500	2
3	3	3 Ford	2019	80000	13500	1
4	4	4 Renault	2017	150000	7500	4
5	5	5 BMW	2021	30000	28500	1

```
INSERT INTO RECENZIE VALUES (recenzie_seq.NEXTVAL, 1, 9, TO_DATE('2023-10-10', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (recenzie_seq.NEXTVAL, 2, 8, TO_DATE('2023-10-11', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (recenzie_seq.NEXTVAL, 3, 7, TO_DATE('2023-10-12', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (recenzie_seq.NEXTVAL, 4, 10, TO_DATE('2023-10-13', 'YYYY-MM-DD'));
INSERT INTO RECENZIE VALUES (recenzie_seq.NEXTVAL, 5, 9, TO_DATE('2023-10-14', 'YYYY-MM-DD'));
```

SELECT * FROM RECENZIE;



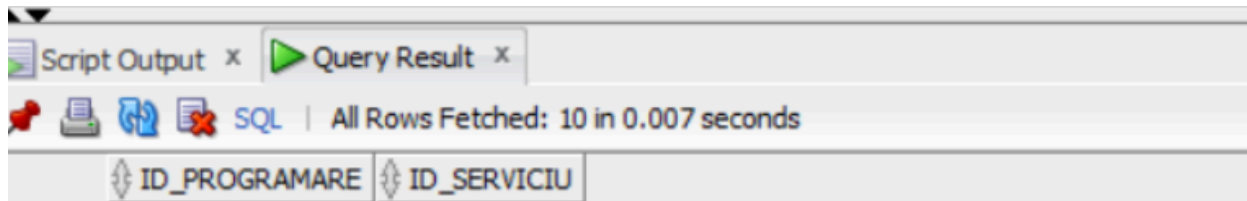
Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.005 seconds

ID_RECENZIE	ID_VEHICUL	NOTA_VEHICUL	DATA_RECENZIE
1	1	1	9 10-OCT-23
2	2	2	8 11-OCT-23
3	3	3	7 12-OCT-23
4	4	4	10 13-OCT-23
5	5	5	9 14-OCT-23

```
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (1, 1);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (1, 2);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (2, 3);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (2, 4);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (3, 5);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (3, 1);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (4, 2);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (4, 3);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (5, 4);
INSERT INTO GESTIONARE_SERVICIU_PROGRAMARE VALUES (5, 5);
```

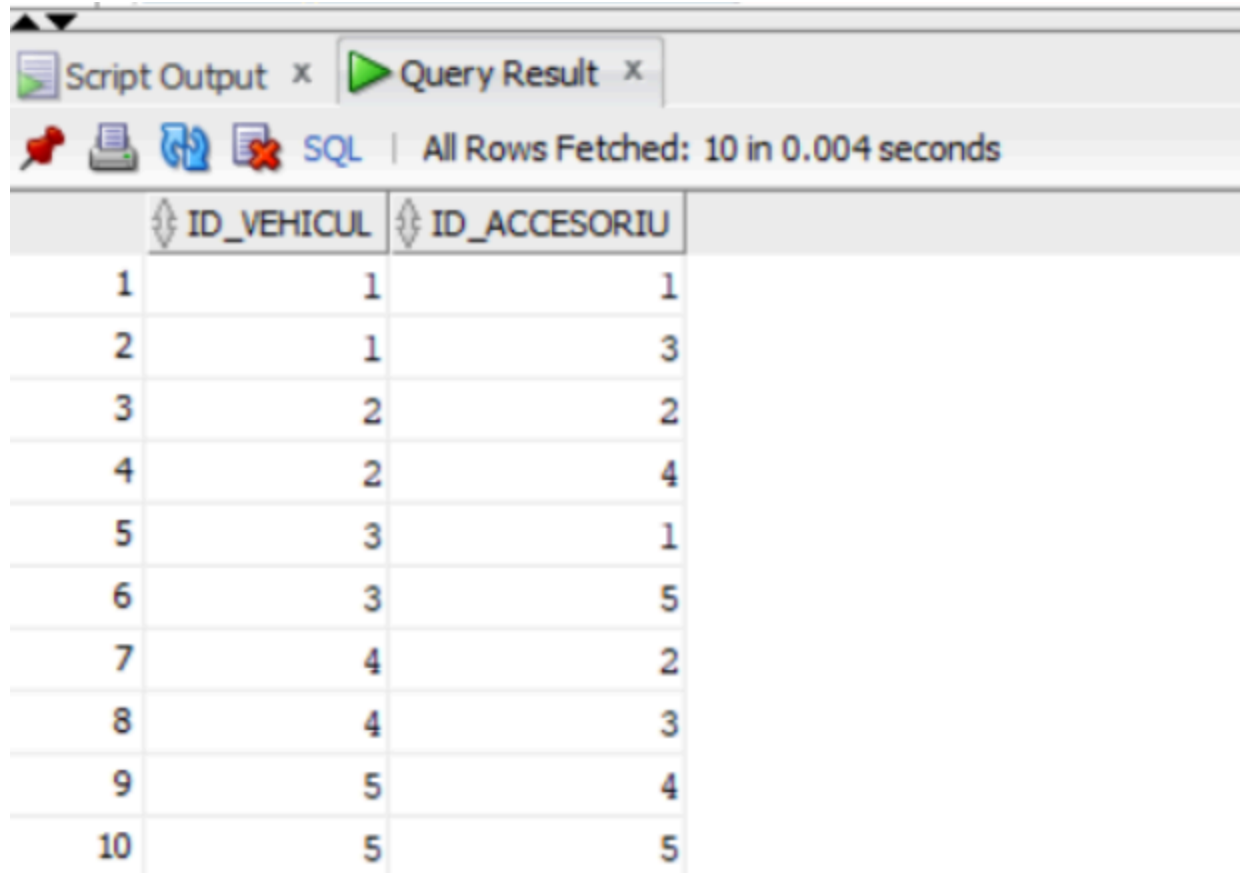
SELECT * FROM GESTIONARE_SERVICIU_PROGRAMARE;



	ID_PROGRAMARE	ID_SERVICIU
1	1	1
2	1	2
3	2	3
4	2	4
5	3	1
6	3	5
7	4	2
8	4	3
9	5	4
10	5	5

```
INSERT INTO DOTARI_EXTRA VALUES (1, 1);  
INSERT INTO DOTARI_EXTRA VALUES (1, 3);  
INSERT INTO DOTARI_EXTRA VALUES (2, 2);  
INSERT INTO DOTARI_EXTRA VALUES (2, 4);  
INSERT INTO DOTARI_EXTRA VALUES (3, 5);  
INSERT INTO DOTARI_EXTRA VALUES (3, 1);  
INSERT INTO DOTARI_EXTRA VALUES (4, 2);  
INSERT INTO DOTARI_EXTRA VALUES (4, 3);  
INSERT INTO DOTARI_EXTRA VALUES (5, 4);  
INSERT INTO DOTARI_EXTRA VALUES (5, 5);
```

```
SELECT * FROM DOTARI_EXTRA;
```



The screenshot shows a database query result window. At the top, there are tabs for 'Script Output' and 'Query Result'. Below the tabs, there are icons for a pin, a printer, a refresh, and a close button, followed by the text 'SQL | All Rows Fetched: 10 in 0.004 seconds'. The main area displays a table with 10 rows and 2 columns: 'ID_VEHICUL' and 'ID_ACCESORIU'. The data is as follows:

	ID_VEHICUL	ID_ACCESORIU
1	1	1
2	1	3
3	2	2
4	2	4
5	3	1
6	3	5
7	4	2
8	4	3
9	5	4
10	5	5

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Problema:

Creați un subprogram stocat care să genereze un raport detaliat pentru un client specificat, incluzând:

- Informații personale (nume, prenume, email);
- Toate serviciile utilizate în programări și prețul lor;
- Toate vehiculele achiziționate (marca și anul de fabricație).

```
CREATE OR REPLACE PROCEDURE raport_client_complet (
    p_id_client CLIENT.id_client%TYPE
)
IS
    -- colectiile folosite
    TYPE tablou_asociativ IS TABLE OF CLIENT%ROWTYPE INDEX BY PLS_INTEGER;
    TYPE nested_table_servicii IS TABLE OF SERVICIU%ROWTYPE;
    TYPE varray_vehicule IS VARRAY(100) OF VEHICUL%ROWTYPE;

    -- variabile
    v_client      tablou_asociativ;
    v_servicii     nested_table_servicii := nested_table_servicii();
    v_vehicule     varray_vehicule := varray_vehicule();
    v_max_id      CLIENT.id_client%TYPE;
BEGIN
    -- id-ul maxim pentru a verifica daca id-ul introdus de la tastatura exista
    SELECT MAX(id_client) INTO v_max_id FROM CLIENT;

    IF p_id_client > v_max_id THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: ID-ul ' || p_id_client || ' depaseste numarul de clienti (' || v_max_id || ').');
        RETURN;
    END IF;

    BEGIN
        SELECT * INTO v_client(1)
        FROM CLIENT
        WHERE id_client = p_id_client;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Eroare: Clientul cu ID ' || p_id_client || ' nu exista. ');
            RETURN;
    END;

    SELECT s.*
    BULK COLLECT INTO v_servicii
    FROM SERVICIU s
    JOIN GESTIONARE_SERVICIU_PROGRAMARE gsp ON s.id_serviciu = gsp.id_serviciu
    JOIN PROGRAMARE p ON gsp.id_programare = p.id_programare
    WHERE p.id_client = p_id_client;

    SELECT v.*
```

```

BULK COLLECT INTO v_vehicule
FROM VEHICUL v
JOIN TRANZACTIE t ON v.id_tranzactie = t.id_tranzactie
WHERE t.id_client = p_id_client;

DBMS_OUTPUT.PUT_LINE('--- RAPORT CLIENT ---');
DBMS_OUTPUT.PUT_LINE('Nume: ' || v_client(1).nume_client || ' ' || v_client(1).prenume_client);
DBMS_OUTPUT.PUT_LINE('Email: ' || v_client(1).adresa_email);

-- afisare servicii
DBMS_OUTPUT.PUT_LINE('--- SERVICII UTILIZATE ---');
IF v_servicii.COUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista servicii asociate pentru acest client. ');
ELSE
    FOR i IN 1..v_servicii.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(v_servicii(i).tip_serviciu || ' - ' || v_servicii(i).pret_serviciu || ' RON');
    END LOOP;
END IF;

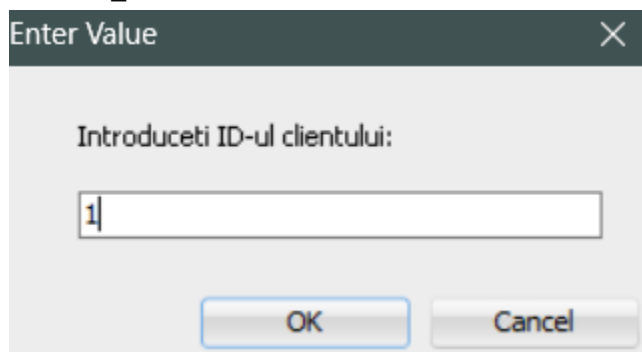
-- afisare vehicule
DBMS_OUTPUT.PUT_LINE('--- VEHICULE ACHIZITIONATE ---');
IF v_vehicule.COUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista vehicule achizitionate de acest client. ');
ELSE
    FOR i IN 1..v_vehicule.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(v_vehicule(i).marca_vehicul || ' (' || v_vehicule(i).an_fabricatie || ')');
    END LOOP;
END IF;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare neasteptata: ' || SQLERRM);
END;
/

```

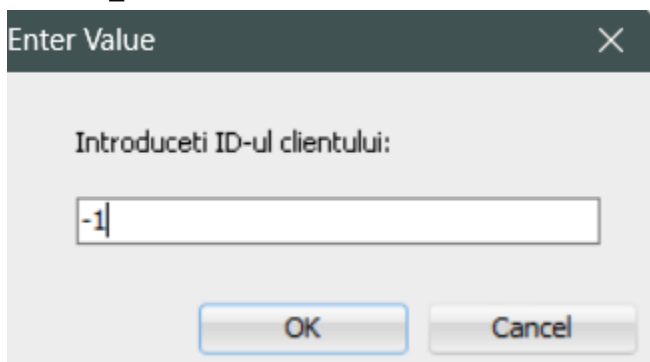
Programul va primi de la tastatură id-ul unui client și va afișa astfel:

1. Pentru id_client = 1:



```
--- RAPORT CLIENT ---  
Nume si prenume: Mihai George  
Email: mihai.george@gmail.com  
--- SERVICII UTILIZATE ---  
Schimb ulei - 150 RON  
Revizie generala - 500 RON  
--- VEHICULE ACHIZITIONATE ---  
Dacia (2018)
```

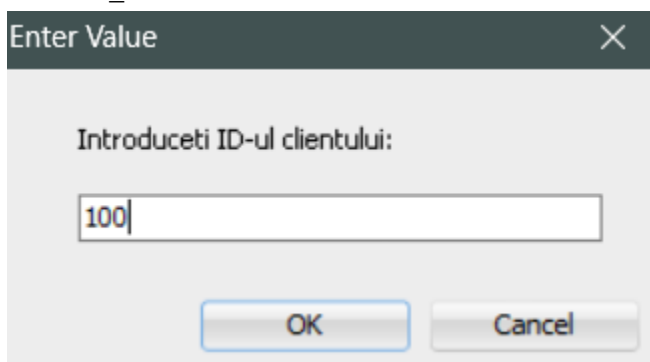
2. Pentru $id_client < 1$:



Eroare: Clientul cu ID -1 nu exista.

PL/SQL procedure successfully completed.

3. Pentru $id_client > 5$:



Eroare: ID-ul 100 depaseste numarul de clienti (5).

PL/SQL procedure successfully completed.

7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

Problema:

Creați un subprogram stocat care să afișeze pentru fiecare client:

- Informații de bază pentru client (nume, prenume, email);
- Toate tranzacțiile efectuate (data și suma totală);
- Vehiculele achiziționate la fiecare tranzacție (marca și anul de fabricație).

```
CREATE OR REPLACE PROCEDURE client_transaction_report
IS
    -- cursor pentru clienti
    CURSOR c_client IS
        SELECT id_client, nume_client, prenume_client, adresa_email
        FROM CLIENT;

    -- cursor parametrizat pentru tranzactii si vehicule, dependent de c_client
    CURSOR c_transaction (p_id_client CLIENT.id_client%TYPE) IS
        SELECT t.id_tranzactie, t.data_tranzactie, t.suma_totala,
               v.marca_vehicul, v.an_fabricatie
        FROM TRANZACTIE t
        JOIN VEHICUL v ON t.id_tranzactie = v.id_tranzactie
        WHERE t.id_client = p_id_client;
BEGIN
    FOR client_rec IN c_client LOOP

        DBMS_OUTPUT.PUT_LINE('Client: ' || client_rec.nume_client || ' ' || client_rec.prenume_client);
        DBMS_OUTPUT.PUT_LINE('Email: ' || client_rec.adresa_email);
        DBMS_OUTPUT.PUT_LINE('-----');

        --se parcurg tranzactiile si vehiculele clientului curent
        FOR trans_rec IN c_transaction(client_rec.id_client) LOOP
            DBMS_OUTPUT.PUT_LINE('Tranzactie ID: ' || trans_rec.id_tranzactie);
            DBMS_OUTPUT.PUT_LINE('Data: ' || TO_CHAR(trans_rec.data_tranzactie, 'DD-MM-YYYY'));
            DBMS_OUTPUT.PUT_LINE('Suma: ' || trans_rec.suma_totala || ' RON');
            DBMS_OUTPUT.PUT_LINE('Vehicul achizitionat: ' || trans_rec.marca_vehicul || ' (' || trans_rec.an_fabricatie || ')');
            DBMS_OUTPUT.PUT_LINE('-----');
        END LOOP;

        DBMS_OUTPUT.NEW_LINE;
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;
/

BEGIN
    client_transaction_report();
```

Maftai Ștefan, grupa 342

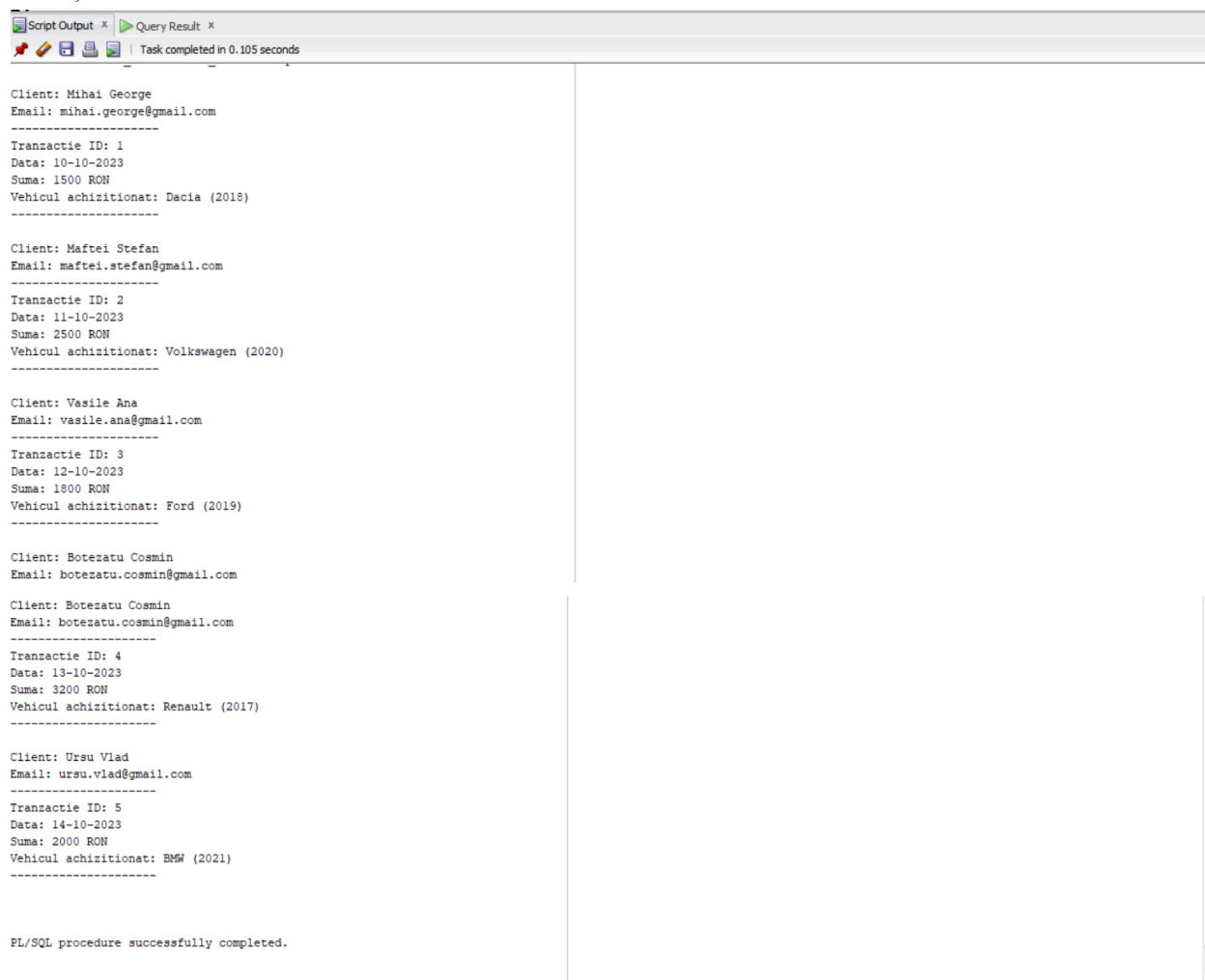
Proiect SGBD

END;

Se utilizeaza:

- Un cursor pentru parcurgerea clienților;
- Un cursor parametrizat care este dependent de cursorul client pentru a prelua tranzacțiile și vehiculele asociate.

Se afișează:



```
Client: Mihai George
Email: mihai.george@gmail.com
-----
Tranzactie ID: 1
Data: 10-10-2023
Suma: 1500 RON
Vehicul achizitionat: Dacia (2018)
-----

Client: Maftai Stefan
Email: maftai.stefan@gmail.com
-----
Tranzactie ID: 2
Data: 11-10-2023
Suma: 2500 RON
Vehicul achizitionat: Volkswagen (2020)
-----

Client: Vasile Ana
Email: vasile.ana@gmail.com
-----
Tranzactie ID: 3
Data: 12-10-2023
Suma: 1800 RON
Vehicul achizitionat: Ford (2019)
-----

Client: Botezatu Cosmin
Email: botezatu.cosmin@gmail.com
-----
Tranzactie ID: 4
Data: 13-10-2023
Suma: 3200 RON
Vehicul achizitionat: Renault (2017)
-----

Client: Ursu Vlad
Email: ursu.vlad@gmail.com
-----
Tranzactie ID: 5
Data: 14-10-2023
Suma: 2000 RON
Vehicul achizitionat: BMW (2021)
-----

PL/SQL procedure successfully completed.
```

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

Problema:

Creați o funcție care calculează suma totală a prețurilor vehiculelor vândute de un manager specificat.

Tabele sunt următoarele:

- Manager
- Tranzactie
- Vehicul

Cele două excepții:

- NO_DATA_FOUND: Dacă managerul nu există sau nu are tranzacții;
- TOO_MANY_ROWS: Dacă interogarea returnează mai multe rezultate decât este așteptat(de exemplu, dacă un vehicul este asociat la mai multe tranzacții în mod neașteptat).

```
CREATE OR REPLACE FUNCTION suma_pret_vehicule_manager (
    p_id_manager MANAGER.id_manager%TYPE
) RETURN NUMBER
IS
    v_pret_vehicul VEHICUL.pret%TYPE;
    v_manager_exists NUMBER;
BEGIN

    SELECT COUNT(*) INTO v_manager_exists
    FROM MANAGER
    WHERE id_manager = p_id_manager;

    IF v_manager_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Managerul cu ID ' || p_id_manager || ' nu există.');
```

-- fara sum() pentru a scoate too_many_rows

```
    END IF;

    SELECT V.pret
    INTO v_pret_vehicul
    FROM TRANZACTIE T
    JOIN VEHICUL V ON T.id_tranzactie = V.id_tranzactie
    WHERE T.id_manager = p_id_manager;

    RETURN v_pret_vehicul;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Managerul cu ID ' || p_id_manager || ' nu are tranzacții.');
```

WHEN TOO_MANY_ROWS THEN

```
        RAISE_APPLICATION_ERROR(-20003, 'Interogarea a returnat mai multe rezultate decat era asteptat.');
```

WHEN OTHERS THEN

```
        RAISE;
    END;
/
```

```
-- testarea pentru too_many_rows
DECLARE
    v_suma NUMBER;
BEGIN
    v_suma := suma_tranzactii_manager(1);
    DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;

/

-- valid
DECLARE
    v_suma NUMBER;
BEGIN
    v_suma := suma_tranzactii_manager(2);
    DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;

/

-- testarea no_data_found caz 1 manager neexistent
DECLARE
    v_suma NUMBER;
BEGIN
    v_suma := suma_pret_vehicule_manager(99);
    DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;

/

-- manager valid fara tranzactii
DECLARE
    v_suma NUMBER;
BEGIN
    v_suma := suma_pret_vehicule_manager(6);
    DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;

/
```

Maftai Ștefan, grupa 342

Proiect SGBD

Test Case 1: too_many_rows

```
-- testarea pentru too_many_rows
DECLARE
  v_suma NUMBER;
BEGIN
  v_suma := suma_tranzactii_manager(1);
  DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;
```

Task completed in 0.056 seconds

Function SUMA_PRET_VEHICULE_MANAGER compiled

Eroare: ORA-20002: Interogarea a returnat mai multe rezultate decât era așteptat.

PL/SQL procedure successfully completed.

Test Case 2: valid

```
-- valid
DECLARE
  v_suma NUMBER;
BEGIN
  v_suma := suma_tranzactii_manager(2);
  DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;
```

Task completed in 0.134 seconds

PL/SQL procedure successfully completed.

Suma totală: 18500 RON

PL/SQL procedure successfully completed.

Test Case 3: no_data_found caz 1 manager neexistent

```
-- testarea no_data_found caz 1 manager neexistent
DECLARE
  v_suma NUMBER;
BEGIN
  v_suma := suma_pret_vehicule_manager(99);
  DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;
```

Task completed in 0.056 seconds

PL/SQL procedure successfully completed.

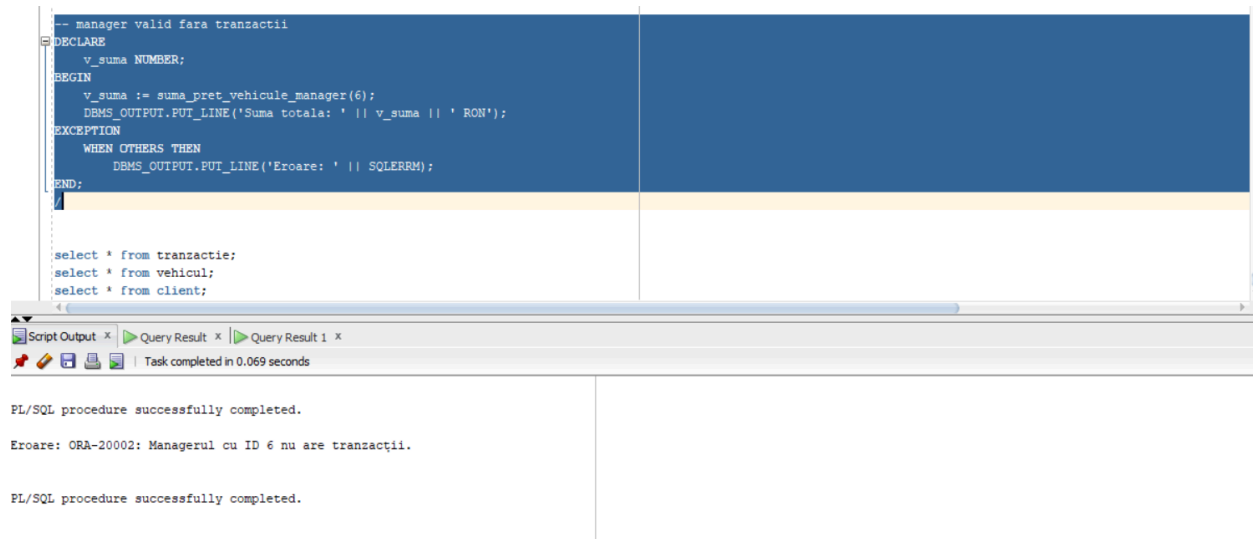
Eroare: ORA-20001: Managerul cu ID 99 nu există.

PL/SQL procedure successfully completed.

Test Case 4: manager valid fara tranzactii

```
-- manager valid fara tranzactii
DECLARE
  v_suma NUMBER;
BEGIN
  v_suma := suma_pret_vehicule_manager(6);
  DBMS_OUTPUT.PUT_LINE('Suma totală: ' || v_suma || ' RON');
EXCEPTION
```


Maftai Ștefan, grupa 342
Proiect SGBD



The screenshot displays the Oracle SQL Developer environment. The top pane shows a PL/SQL script with the following code:

```
-- manager valid fara tranzactii
DECLARE
  v_suma NUMBER;
BEGIN
  v_suma := suma_pret_vehicule_manager(6);
  DBMS_OUTPUT.PUT_LINE('Suma totala: ' || v_suma || ' RON');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;
```

Below the script, three SQL queries are listed:

```
select * from tranzactie;
select * from vehicul;
select * from client;
```

The bottom pane shows the execution results. It includes a status bar indicating "Task completed in 0.069 seconds" and a message area with the following text:

PL/SQL procedure successfully completed.

Eroare: ORA-20002: Managerul cu ID 6 nu are tranzactii.

PL/SQL procedure successfully completed.

9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate

Problema:

Creați o procedură care afișează toate programările într-un interval de timp specificat, care vor include: detalii despre programări, serviciile asociate, angajații care au realizat serviciile, suma totală cheltuită. Procedura va utiliza 5 tabele acestea fiind Client, Programare, Gestionare_Serviciu_Programare, Serviciu și Angajat.

Excepțiile sunt:

- Client inexistent.
- Nu există programări în intervalul specificat.

```
CREATE OR REPLACE PROCEDURE raport_programari_client (
    p_id_client CLIENT.id_client%TYPE,
    p_data_start DATE,
    p_data_end DATE
)
IS
    -- excepțiile custom
    client_inexistent EXCEPTION;
    PRAGMA EXCEPTION_INIT(client_inexistent, -20010);
    programari_inexistente EXCEPTION;
    PRAGMA EXCEPTION_INIT(programari_inexistente, -20011);

    v_client_exists NUMBER;
    v_programari_count NUMBER;
    v_suma_totala NUMBER := 0;
BEGIN
    SELECT COUNT(*) INTO v_client_exists FROM CLIENT WHERE id_client = p_id_client;
    IF v_client_exists = 0 THEN
        RAISE_APPLICATION_ERROR(-20010, 'Clientul cu ID ' || p_id_client || ' nu exista in sistem');
    END IF;

    SELECT COUNT(*) INTO v_programari_count
    FROM PROGRAMARE
    WHERE id_client = p_id_client
    AND data_programarii BETWEEN p_data_start AND p_data_end;

    IF v_programari_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20011, 'Nicio programare gasita pentru clientul ' || p_id_client || ' in perioada
specificata');
    END IF;

    FOR rec IN (
        SELECT
            c.nume_client,
            p.data_programarii,
```

```

        p.starea_programarii,
        s.tip_serviciu,
        s.pret_serviciu,
        m.nume_manager,
        m.prenume_manager,
        t.suma_totala
    FROM CLIENT c
    JOIN PROGRAMARE p ON c.id_client = p.id_client
    JOIN GESTIONARE_SERVICIU_PROGRAMARE gsp ON p.id_programare = gsp.id_programare
    JOIN SERVICIU s ON gsp.id_serviciu = s.id_serviciu
    JOIN TRANZACTIE t ON p.id_client = t.id_client
    JOIN MANAGER m ON t.id_manager = m.id_manager
    WHERE c.id_client = p_id_client
    AND p.data_programarii BETWEEN p_data_start AND p_data_end
) LOOP
    v_suma_totala := v_suma_totala + rec.pret_serviciu;

    DBMS_OUTPUT.PUT_LINE('Client: ' || rec.nume_client);
    DBMS_OUTPUT.PUT_LINE('Data programarii: ' || TO_CHAR(rec.data_programarii, 'DD-MM-YYYY'));
    DBMS_OUTPUT.PUT_LINE('Stare: ' || rec.starea_programarii);
    DBMS_OUTPUT.PUT_LINE('Serviciu: ' || rec.tip_serviciu || ' (' || rec.pret_serviciu || ' RON)');
    DBMS_OUTPUT.PUT_LINE('Manager responsabil: ' || rec.nume_manager || ' ' || rec.prenume_manager);
    DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;

DBMS_OUTPUT.PUT_LINE('Suma totala cheltuita: ' || v_suma_totala || ' RON');

EXCEPTION
    WHEN client_inexistent THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: Clientul specificat nu exista in baza de date');
    WHEN programari_inexistente THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: Nu exista programari in intervalul specificat');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare neasteptata: ' || SQLCODE || ' - ' || SQLERRM);
END;
/

BEGIN
    -- client existent
    raport_programari_client(1, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-12-31', 'YYYY-MM-DD'));

    -- client neexistent
    raport_programari_client(999, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-12-31', 'YYYY-MM-DD'));

    -- fara programari
    raport_programari_client(1, TO_DATE('2030-01-01', 'YYYY-MM-DD'), TO_DATE('2030-12-31', 'YYYY-MM-DD'));
END;
/

```

Maftei Ștefan, grupa 342

Proiect SGBD

The image displays three sequential screenshots of the Oracle SQL Developer environment, showing the execution of a PL/SQL procedure and subsequent queries.

First Screenshot: The SQL script editor shows a PL/SQL procedure named `raport_programari_client` with parameters `p_id_client => 1`, `p_data_start => TO_DATE('2023-01-01', 'YYYY-MM-DD')`, and `p_data_end => TO_DATE('2023-12-31', 'YYYY-MM-DD')`. The script is executed, and the output window shows the procedure completed successfully. The output displays client information for Mihai, including program date (10-10-2023), status (Confirmata), service (Schimb ulei (150 RON)), and employee (Munteanu Vlad). It also shows a total cost of 650 RON.

Second Screenshot: The SQL script editor shows the same procedure, but with `p_id_client => 123`. The output window shows the procedure completed successfully, but then displays an error: `Eroare: ORA-20010: Clientul cu ID 123 nu exista.` (Error: ORA-20010: Client with ID 123 does not exist).

Third Screenshot: The SQL script editor shows the same procedure, but with `p_id_client => 1` and `p_data_start => TO_DATE('2024-01-01', 'YYYY-MM-DD')`. The output window shows the procedure completed successfully, but then displays an error: `Eroare: ORA-20011: Nu există programări între 01-01-2024 și 31-12-2024.` (Error: ORA-20011: No bookings between 01-01-2024 and 31-12-2024).

The bottom of the interface shows the `Compiler - Log` window, which is currently empty.

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

-- trigger la nivel de comanda

```
CREATE TABLE audit_tranzactii (  
    id_audit NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    utilizator VARCHAR2(100),  
    tip_operatie VARCHAR2(10),  
    data_operatie DATE,  
    detalii VARCHAR2(500)  
);
```

```
CREATE OR REPLACE TRIGGER trg_audit_tranzactii  
AFTER INSERT OR UPDATE OR DELETE ON TRANZACTIE  
FOR EACH ROW  
DECLARE  
    v_tip_operatie VARCHAR2(10);  
BEGIN
```

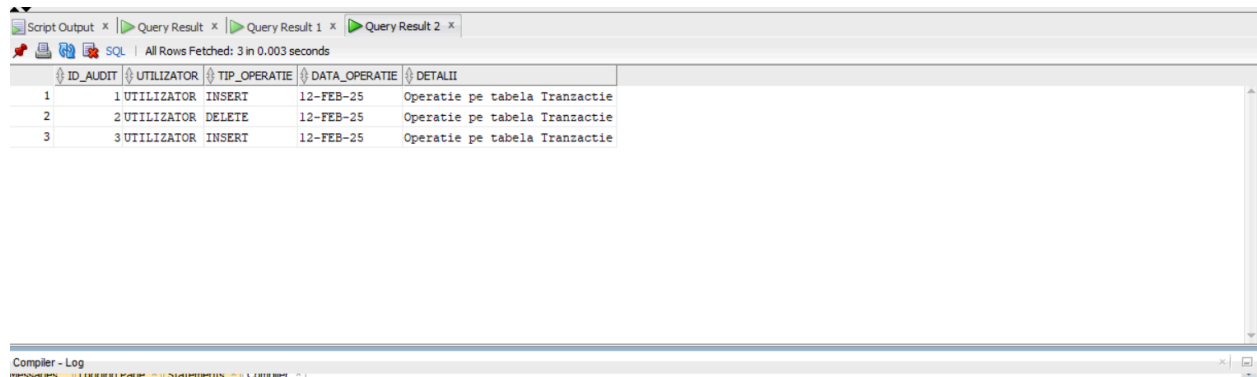
Maftei Ștefan, grupa 342
Proiect SGBD

```
IF INSERTING THEN
    v_tip_operatie := 'INSERT';
ELSIF UPDATING THEN
    v_tip_operatie := 'UPDATE';
ELSIF DELETING THEN
    v_tip_operatie := 'DELETE';
END IF;
```

```
INSERT INTO audit_tranzactii (
    utilizator,
    tip_operatie,
    data_operatie,
    detalii
) VALUES (
    USER,
    v_tip_operatie,
    SYSDATE,
    'Operatie pe tabela Tranzactie'
);
END;
/

INSERT INTO TRANZACTIE (id_tranzactie, id_client, id_manager, data_tranzactie, suma_totala) VALUES
(tranzactie_seq.NEXTVAL, 1, 1, TO_DATE('2021-03-27', 'YYYY-MM-DD'), 5000);
COMMIT;
DELETE FROM TRANZACTIE WHERE id_tranzactie = 21;
COMMIT;

SELECT * FROM audit_tranzactii;
```



ID_AUDIT	UTILIZATOR	TIP_OPERATIE	DATA_OPERATIE	DETALII
1	1 UTILIZATOR	INSERT	12-FEB-25	Operatie pe tabela Tranzactie
2	2 UTILIZATOR	DELETE	12-FEB-25	Operatie pe tabela Tranzactie
3	3 UTILIZATOR	INSERT	12-FEB-25	Operatie pe tabela Tranzactie

Compiler - Log

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

-- trigger la nivel de linie

```
CREATE TABLE Audit_Salarii (  
    ID_Audit NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    ID_Angajat NUMBER NOT NULL,  
    Salariu_Vechi NUMBER,  
    Salariu_Nou NUMBER,  
    Data_Modificare DATE,  
    Utilizator VARCHAR2(100)  
);
```

```
CREATE OR REPLACE TRIGGER audit_salarii_trigger  
AFTER UPDATE OF salariu_angajat ON angajat  
FOR EACH ROW  
BEGIN  
    INSERT INTO audit_salarii (  
        id_angajat,  
        salariu_vechi,  
        salariu_nou,  
        data_modificare,  
        utilizator  
    ) VALUES (  
        :NEW.id_angajat,  
        :OLD.salariu_angajat,  
        :NEW.salariu_angajat,  
        SYSDATE,  
        USER
```

Maftei Ștefan, grupa 342
Proiect SGBD

```
:OLD.id_angajat,  
:OLD.salariu_angajat,  
:NEW.salariu_angajat,  
SYSDATE,  
USER  
);  
END;  
/
```

```
UPDATE angajat  
SET salariu_angajat = 5000  
WHERE id_angajat = 1;  
COMMIT;  
SELECT * FROM audit_salarii;
```

ID_AUDIT	ID_ANGAJAT	SALARIU_VECHI	SALARIU_NOU	DATA_MODIFICARE	UTILIZATOR
1	1	4500	5000	12-FEB-25	UTILIZATOR

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

```
-- ldd
```

```
CREATE TABLE audit_ldd (  
    id_audit    NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
    utilizator  VARCHAR2(100),  
    tip_operatie VARCHAR2(50),  
    obiect     VARCHAR2(100),  
    data_operatie DATE  
);
```

```
CREATE OR REPLACE TRIGGER trg_audit_ldd  
AFTER CREATE OR ALTER OR DROP ON DATABASE  
DECLARE  
    v_tip_operatie VARCHAR2(50);  
    v_obiect       VARCHAR2(100);  
BEGIN
```

```
    v_tip_operatie := ORA_SYSEVENT;  
    v_obiect       := ORA_DICT_OBJ_NAME;
```

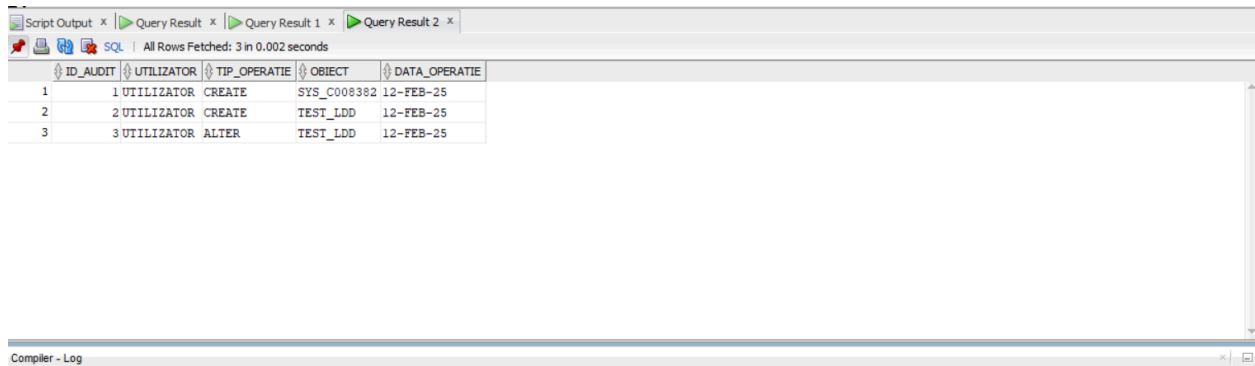
```
INSERT INTO audit_ldd (  

```


Maftei Ștefan, grupa 342
Proiect SGBD

```
    utilizator,  
    tip_operatie,  
    obiect,  
    data_operatie  
  ) VALUES (  
    USER,  
    v_tip_operatie,  
    v_obiect,  
    SYSDATE  
  );  
END;  
/
```

```
CREATE TABLE test_ldd (  
  id NUMBER PRIMARY KEY,  
  nume VARCHAR2(50)  
);  
ALTER TABLE test_ldd ADD (descriere VARCHAR2(200));  
SELECT * FROM audit_ldd;
```



ID_AUDIT	UTILIZATOR	TIP_OPERATIE	OBIECT	DATA_OPERATIE
1	1 UTILIZATOR	CREATE	SYS_C008382	12-FEB-25
2	2 UTILIZATOR	CREATE	TEST_LDD	12-FEB-25
3	3 UTILIZATOR	ALTER	TEST_LDD	12-FEB-25