



# Gestiunea unei Gradini Comunale

## Beldi Darius Vlad

Seria 24 Grupa 242

2024-2025

# Cuprins

Detalii tehnice .....	3
1.   Prezentarea bazei de date .....	4
2.   Diagrama ERD.....	5
3.   Diagrama Conceptuala .....	6
.....	6
4.   Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc) .....	7
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. ....	21
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. ....	24
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. ....	27
10.   Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.....	32
11.   Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.....	34
12.   Definiți un trigger de tip LDD. Declanșați trigger-ul.....	36

## Detalii tehnice

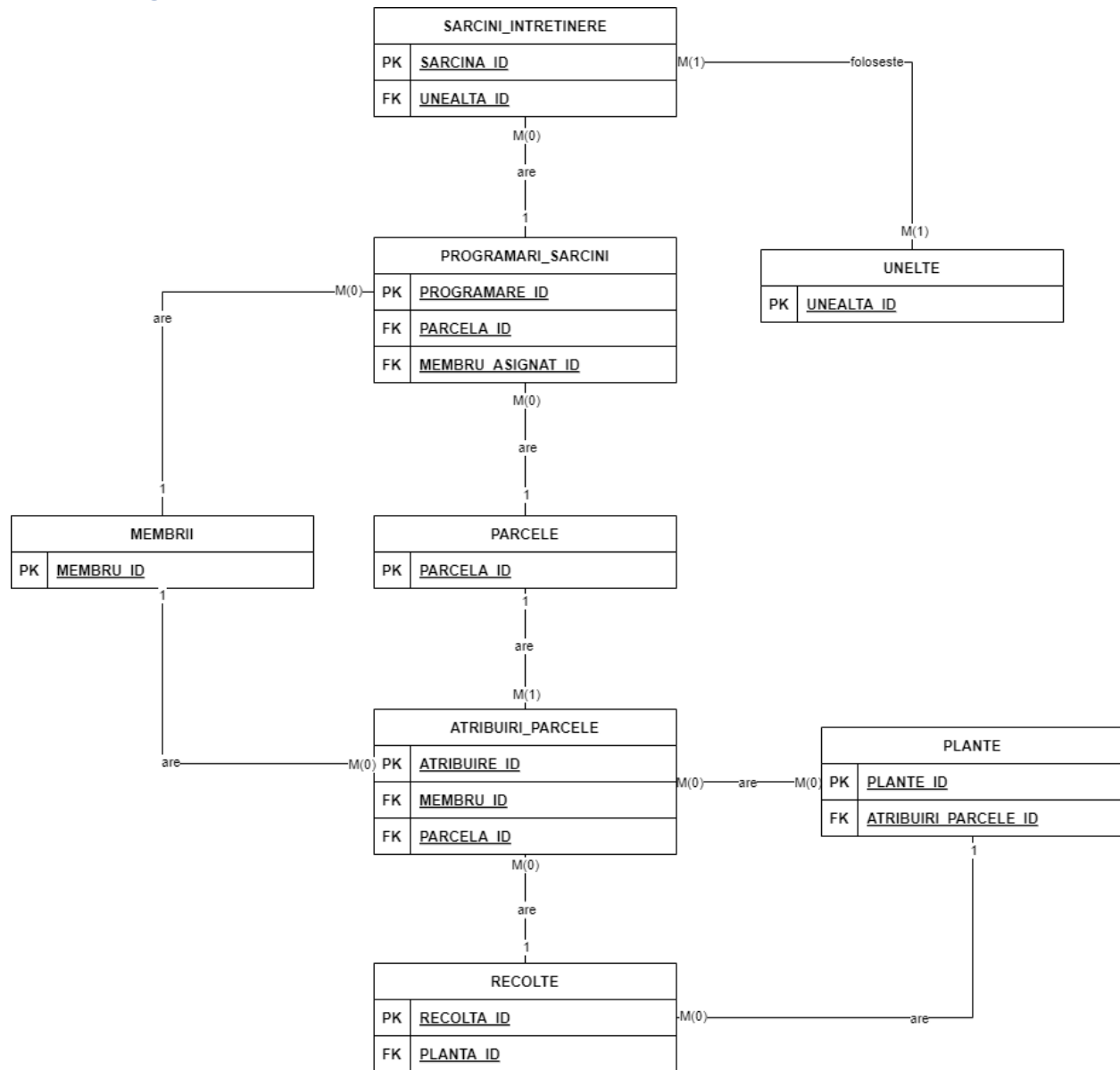
Tema aleasa de mine este: Gestiunea unei Gradini Comunale.

BANNER		
-----		
BANNER_FULL		
-----		
BANNER_LEGACY		
-----		
CON_ID		
-----		
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production		
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production		
Version 21.3.0.0.0		
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production		
0		
BANNER		
-----		
BANNER_FULL		
-----		
BANNER_LEGACY		
-----		
CON_ID		
-----		
NAME		
-----		
VALUE		
-----		
sga_target		PORT_STRING
1610612736		-----
memory_target		IBMPC/WIN_NT64-9.1.0
0		
pga_aggregate_target		
536870912		

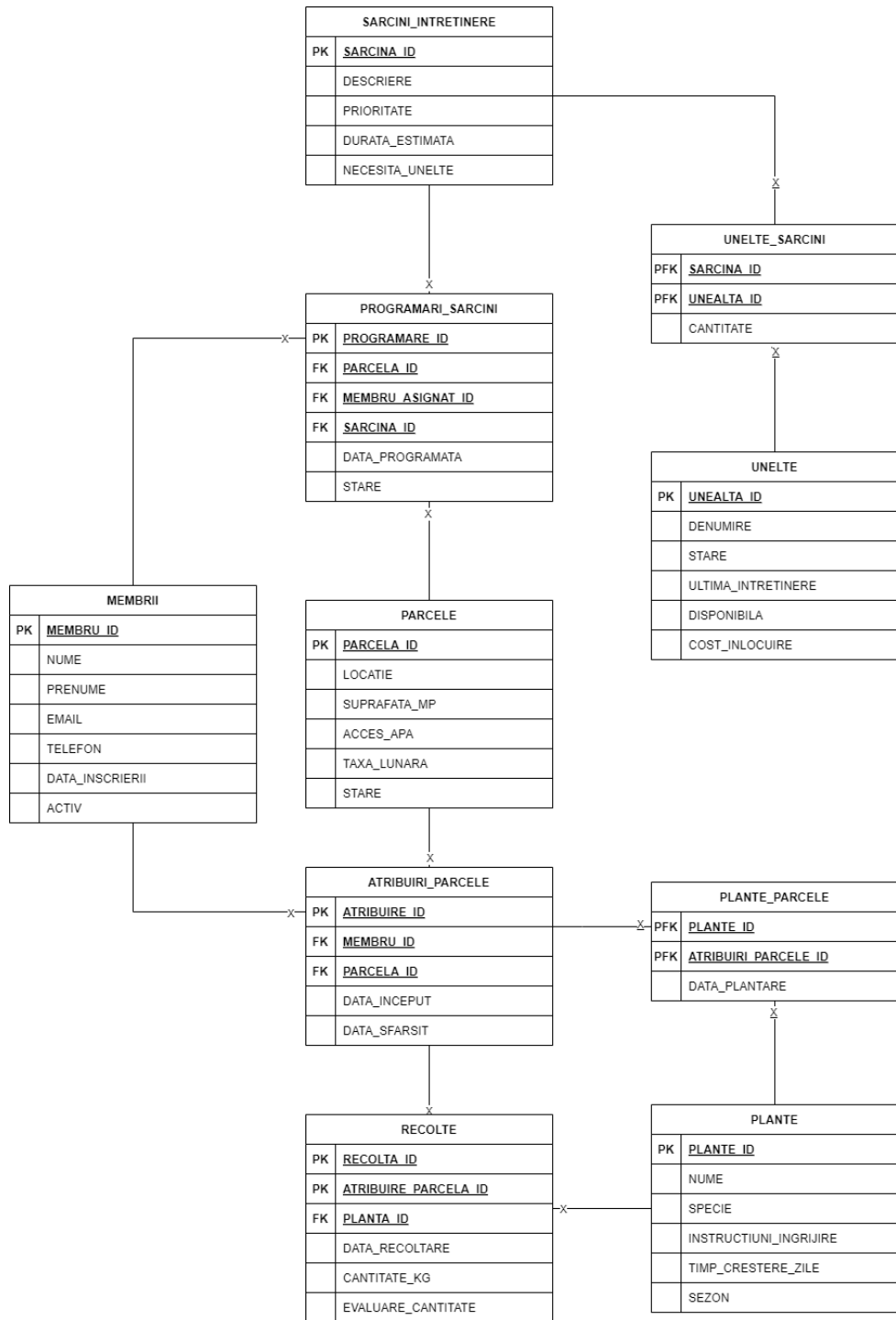
## 1. Prezentarea bazei de date

Baza de date "Gestiunea Gradinii Comunitare" este conceputa pentru a administra o gradina comunitara urbana. Aceasta permite gestionarea eficienta a parcelelor de gradina, a membrilor comunitatii, a plantelor cultivate și a activitatilor de intretinere. Sistemul faciliteaza urmarirea recoltelor, programarea sarcinilor de intretinere și gestiunea uneltelor comune. Este utila pentru asociatiile de gradinarit comunitar, permitand o organizare transparenta a resurselor si activitatilor.

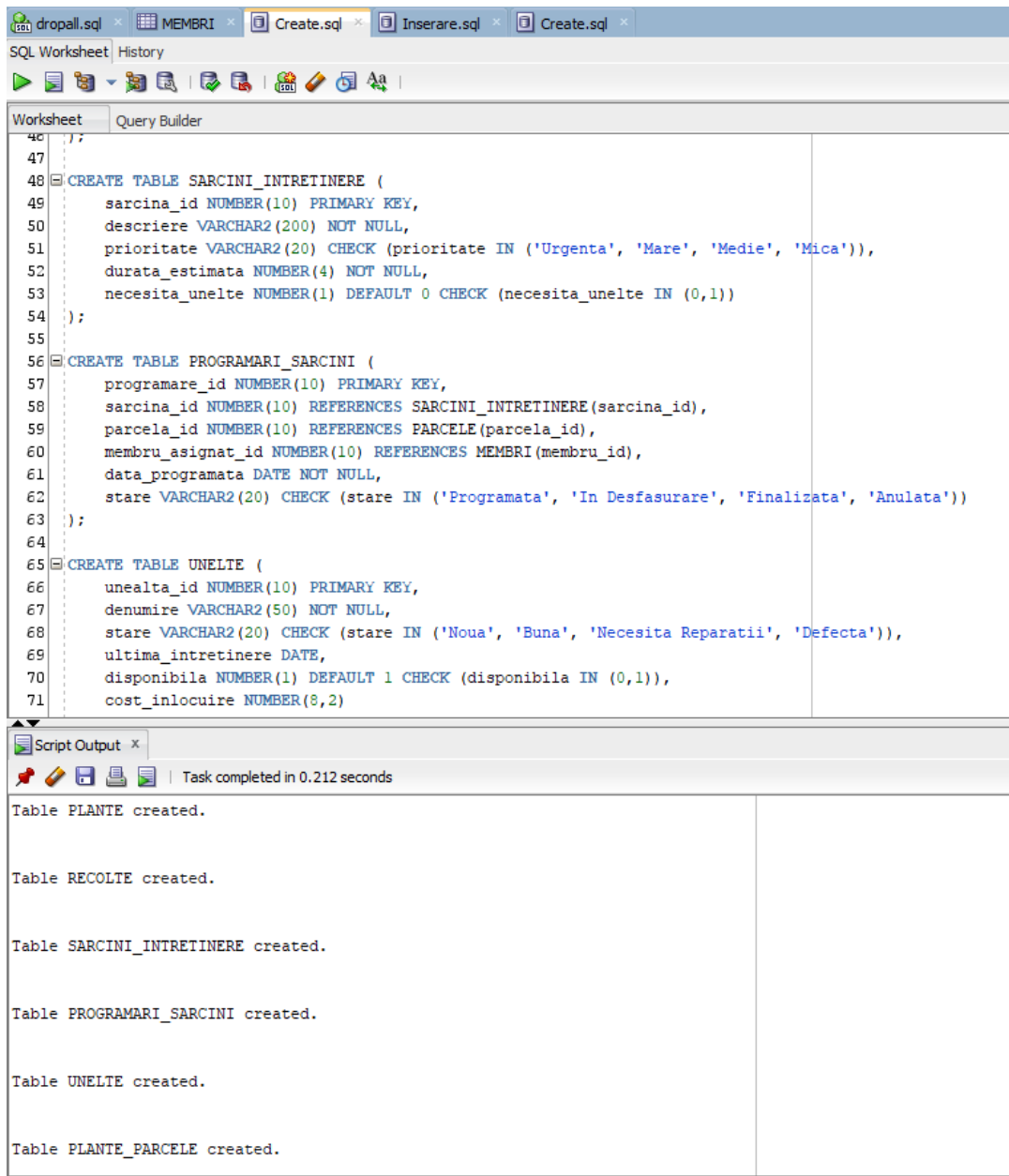
## 2. Diagrama ERD



### 3. Diagrama Conceptuala



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



```

46 ;
47
48 CREATE TABLE SARCINI_INTRETINERE (
49     sarcina_id NUMBER(10) PRIMARY KEY,
50     descriere VARCHAR2(200) NOT NULL,
51     prioritate VARCHAR2(20) CHECK (prioritate IN ('Urgenta', 'Mare', 'Medie', 'Mica')),
52     durata_estimata NUMBER(4) NOT NULL,
53     necesita_unelte NUMBER(1) DEFAULT 0 CHECK (necesita_unelte IN (0,1))
54 );
55
56 CREATE TABLE PROGRAMARI_SARCINI (
57     programare_id NUMBER(10) PRIMARY KEY,
58     sarcina_id NUMBER(10) REFERENCES SARCINI_INTRETINERE(sarcina_id),
59     parcela_id NUMBER(10) REFERENCES PARCELE(parcela_id),
60     membru_asignat_id NUMBER(10) REFERENCES MEMBRI(membru_id),
61     data_programata DATE NOT NULL,
62     stare VARCHAR2(20) CHECK (stare IN ('Programata', 'In Desfasurare', 'Finalizata', 'Anulata'))
63 );
64
65 CREATE TABLE UNELTE (
66     unealta_id NUMBER(10) PRIMARY KEY,
67     denumire VARCHAR2(50) NOT NULL,
68     stare VARCHAR2(20) CHECK (stare IN ('Noua', 'Buna', 'Necesita Reparatii', 'Defecta')),
69     ultima_intretinere DATE,
70     disponibila NUMBER(1) DEFAULT 1 CHECK (disponibila IN (0,1)),
71     cost_inlocuire NUMBER(8,2)

```

Script Output x

Task completed in 0.212 seconds

Table PLANTE created.

Table RECOLTE created.

Table SARCINI\_INTRETINERE created.

Table PROGRAMARI\_SARCINI created.

Table UNELTE created.

Table PLANTE\_PARCELE created.

```
CREATE TABLE MEMBRII (  
    membru_id NUMBER(10) PRIMARY KEY,  
    nume VARCHAR2(50) NOT NULL,  
    prenume VARCHAR2(50) NOT NULL,  
    email VARCHAR2(100) UNIQUE NOT NULL,  
    telefon VARCHAR2(15),  
    data_inscriere DATE DEFAULT SYSDATE NOT NULL,  
    activ NUMBER(1) DEFAULT 1 CHECK (activ IN (0,1))  
);
```

```
CREATE TABLE PARCELE (  
    parcela_id NUMBER(10) PRIMARY KEY,  
    locatie VARCHAR2(100) NOT NULL,  
    suprafata_mp NUMBER(6,2) NOT NULL,  
    acces_apa NUMBER(1) DEFAULT 0 CHECK (acces_apa IN (0,1)),  
    taxa_lunara NUMBER(8,2) NOT NULL,  
    stare VARCHAR2(20) CHECK (stare IN ('Disponibila', 'Ocupata', 'In Mentenanta'))  
);
```

```
CREATE TABLE ATRIBUIRI_PARCELE (  
    atribuire_id NUMBER(10) PRIMARY KEY,  
    membru_id NUMBER(10) REFERENCES MEMBRII(membru_id),  
    parcela_id NUMBER(10) REFERENCES PARCELE(parcela_id),  
    data_inceput DATE NOT NULL,  
    data_sfarsit DATE,  
    CONSTRAINT chk_date CHECK (data_sfarsit > data_inceput)  
);
```



```
CREATE TABLE PLANTE (  
    planta_id NUMBER(10) PRIMARY KEY,  
    nume VARCHAR2(50) NOT NULL,  
    specie VARCHAR2(50) NOT NULL,  
    instructiuni_ingrijire CLOB,  
    timp_crestere_zile NUMBER(4),  
    sezon VARCHAR2(20) CHECK (sezon IN ('Primavara', 'Vara', 'Toamna', 'Iarna', 'Tot anul'))  
);
```

```
CREATE TABLE RECOLTE (  
    recolta_id NUMBER(10) PRIMARY KEY,  
    atribuire_parcela_id NUMBER(10) REFERENCES ATRIBUIRI_PARCELE(atribuire_id),  
    planta_id NUMBER(10) REFERENCES PLANTE(planta_id),  
    data_recoltare DATE NOT NULL,  
    cantitate_kg NUMBER(6,2) NOT NULL,  
    evaluare_calitate VARCHAR2(20) CHECK (evaluare_calitate IN ('Excelenta', 'Buna', 'Medie', 'Slaba'))  
);
```

```
CREATE TABLE SARCINI_INTRETINERE (  
    sarcina_id NUMBER(10) PRIMARY KEY,  
    descriere VARCHAR2(200) NOT NULL,  
    prioritate VARCHAR2(20) CHECK (prioritate IN ('Urgenta', 'Mare', 'Medie', 'Mica')),  
    durata_estimata NUMBER(4) NOT NULL,  
    necesita_unelte NUMBER(1) DEFAULT 0 CHECK (necesita_unelte IN (0,1))  
);
```

```
CREATE TABLE PROGRAMARI_SARCINI (  
    programare_id NUMBER(10) PRIMARY KEY,  
    sarcina_id NUMBER(10) REFERENCES SARCINI_INTRETINERE(sarcina_id),
```

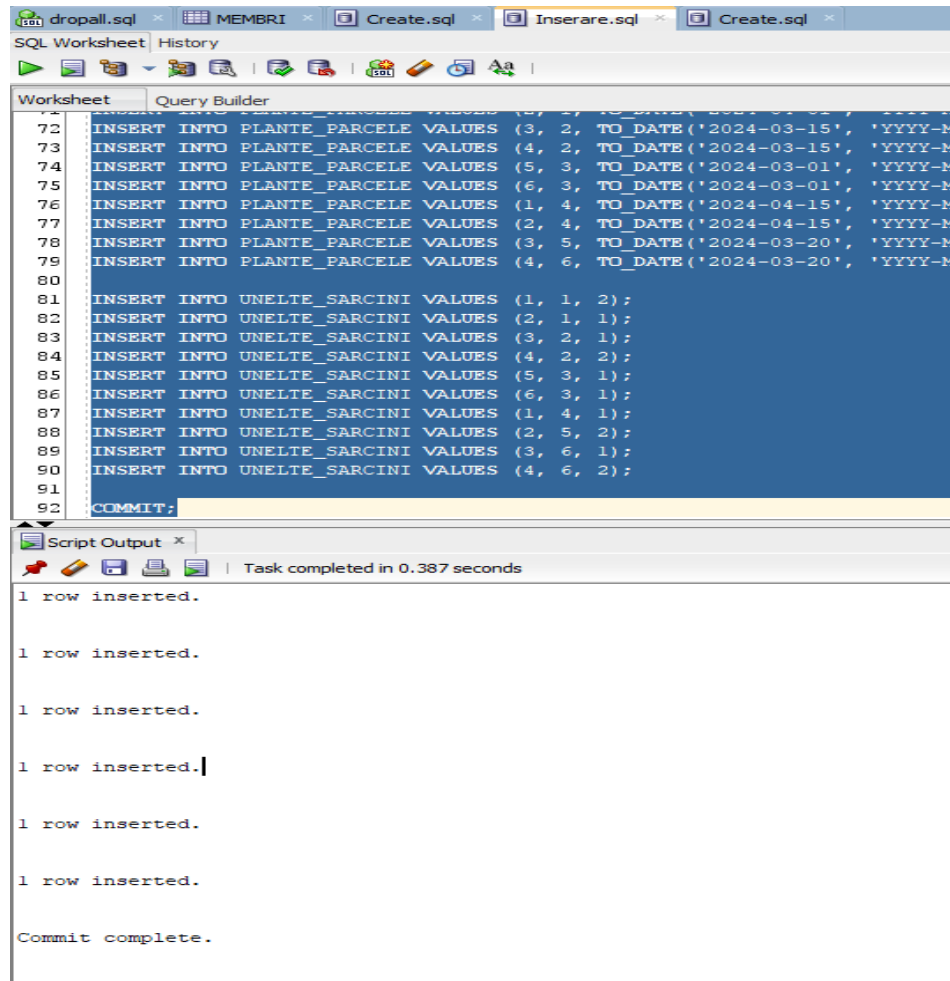
```
parcela_id NUMBER(10) REFERENCES PARCELE(parcela_id),
membru_asignat_id NUMBER(10) REFERENCES MEMBRII(membru_id),
data_programata DATE NOT NULL,
stare VARCHAR2(20) CHECK (stare IN ('Programata', 'In Desfasurare', 'Finalizata', 'Anulata'))
);
```

```
CREATE TABLE UNELTE (
    unealta_id NUMBER(10) PRIMARY KEY,
    denumire VARCHAR2(50) NOT NULL,
    stare VARCHAR2(20) CHECK (stare IN ('Noua', 'Buna', 'Necesita Reparatii', 'Defecta')),
    ultima_intretinere DATE,
    disponibila NUMBER(1) DEFAULT 1 CHECK (disponibila IN (0,1)),
    cost_inlocuire NUMBER(8,2)
);
```

```
CREATE TABLE PLANTE_PARCELE (
    planta_id NUMBER(10) REFERENCES PLANTE(planta_id),
    atribuire_parcela_id NUMBER(10) REFERENCES ATRIBUIRI_PARCELE(atribuire_id),
    data_plantare DATE NOT NULL,
    CONSTRAINT pk_plante_parcele PRIMARY KEY (planta_id, atribuire_parcela_id)
);
```

```
CREATE TABLE UNELTE_SARCINI (
    unealta_id NUMBER(10) REFERENCES UNELTE(unealta_id),
    sarcina_id NUMBER(10) REFERENCES SARCINI_INTRETINERE(sarcina_id),
    cantitate NUMBER(2) DEFAULT 1,
    CONSTRAINT pk_unelte_sarcini PRIMARY KEY (unealta_id, sarcina_id)
);
```

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



```
dropall.sql MEMBRI Create.sql Inserare.sql Create.sql
SQL Worksheet History
Worksheet Query Builder
72 INSERT INTO PLANTE_PARCELE VALUES (3, 2, TO_DATE('2024-03-15', 'YYYY-MM-DD'), TO_DATE('2024-03-15', 'YYYY-MM-DD'));
73 INSERT INTO PLANTE_PARCELE VALUES (4, 2, TO_DATE('2024-03-15', 'YYYY-MM-DD'), TO_DATE('2024-03-15', 'YYYY-MM-DD'));
74 INSERT INTO PLANTE_PARCELE VALUES (5, 3, TO_DATE('2024-03-01', 'YYYY-MM-DD'), TO_DATE('2024-03-01', 'YYYY-MM-DD'));
75 INSERT INTO PLANTE_PARCELE VALUES (6, 3, TO_DATE('2024-03-01', 'YYYY-MM-DD'), TO_DATE('2024-03-01', 'YYYY-MM-DD'));
76 INSERT INTO PLANTE_PARCELE VALUES (1, 4, TO_DATE('2024-04-15', 'YYYY-MM-DD'), TO_DATE('2024-04-15', 'YYYY-MM-DD'));
77 INSERT INTO PLANTE_PARCELE VALUES (2, 4, TO_DATE('2024-04-15', 'YYYY-MM-DD'), TO_DATE('2024-04-15', 'YYYY-MM-DD'));
78 INSERT INTO PLANTE_PARCELE VALUES (3, 5, TO_DATE('2024-03-20', 'YYYY-MM-DD'), TO_DATE('2024-03-20', 'YYYY-MM-DD'));
79 INSERT INTO PLANTE_PARCELE VALUES (4, 6, TO_DATE('2024-03-20', 'YYYY-MM-DD'), TO_DATE('2024-03-20', 'YYYY-MM-DD'));
80
81 INSERT INTO UNELTE_SARCINI VALUES (1, 1, 2);
82 INSERT INTO UNELTE_SARCINI VALUES (2, 1, 1);
83 INSERT INTO UNELTE_SARCINI VALUES (3, 2, 1);
84 INSERT INTO UNELTE_SARCINI VALUES (4, 2, 2);
85 INSERT INTO UNELTE_SARCINI VALUES (5, 3, 1);
86 INSERT INTO UNELTE_SARCINI VALUES (6, 3, 1);
87 INSERT INTO UNELTE_SARCINI VALUES (1, 4, 1);
88 INSERT INTO UNELTE_SARCINI VALUES (2, 5, 2);
89 INSERT INTO UNELTE_SARCINI VALUES (3, 6, 1);
90 INSERT INTO UNELTE_SARCINI VALUES (4, 6, 2);
91
92 COMMIT;
```

```
Script Output x
Task completed in 0.387 seconds
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.
```

```
CREATE SEQUENCE seq_membrii START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE seq_parcele START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE seq_atribuiri START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE seq_plante START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE seq_recolte START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE seq_sarcini START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE seq_programari START WITH 1 INCREMENT BY 1;
```

```
CREATE SEQUENCE seq_unelte START WITH 1 INCREMENT BY 1;
```

```
INSERT INTO membrii VALUES (seq_membrii.NEXTVAL, 'Popescu', 'Ion', 'popescu.ion@email.com',  
'0722111222', TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1);
```

```
INSERT INTO membrii VALUES (seq_membrii.NEXTVAL, 'Ionescu', 'Maria', 'ionescu.maria@email.com',  
'0733222333', TO_DATE('2024-01-02', 'YYYY-MM-DD'), 1);
```

```
INSERT INTO membrii VALUES (seq_membrii.NEXTVAL, 'Popa', 'Ana', 'popa.ana@email.com',  
'0744333444', TO_DATE('2024-01-03', 'YYYY-MM-DD'), 1);
```

```
INSERT INTO membrii VALUES (seq_membrii.NEXTVAL, 'Dumitrescu', 'Vasile',  
'dumitrescu.vasile@email.com', '0755444555', TO_DATE('2024-01-04', 'YYYY-MM-DD'), 1);
```

```
INSERT INTO membrii VALUES (seq_membrii.NEXTVAL, 'Stanescu', 'Elena', 'stanescu.elena@email.com',  
'0766555666', TO_DATE('2024-01-05', 'YYYY-MM-DD'), 1);
```

```
INSERT INTO membrii VALUES (seq_membrii.NEXTVAL, 'Georgescu', 'Andrei',  
'georgescu.andrei@email.com', '0777666777', TO_DATE('2024-01-06', 'YYYY-MM-DD'), 1);
```

```
INSERT INTO PARCELE VALUES (seq_parcele.NEXTVAL, 'Zona A1', 50.00, 1, 100.00, 'Ocupata');
```

```
INSERT INTO PARCELE VALUES (seq_parcele.NEXTVAL, 'Zona A2', 45.00, 1, 90.00, 'Ocupata');
```

```
INSERT INTO PARCELE VALUES (seq_parcele.NEXTVAL, 'Zona B1', 60.00, 1, 120.00, 'Ocupata');
```

```
INSERT INTO PARCELE VALUES (seq_parcele.NEXTVAL, 'Zona B2', 55.00, 1, 110.00, 'In Mentenanta');
```

```
INSERT INTO PARCELE VALUES (seq_parcele.NEXTVAL, 'Zona C1', 40.00, 0, 80.00, 'Disponibila');
```

```
INSERT INTO PARCELE VALUES (seq_parcele.NEXTVAL, 'Zona C2', 48.00, 1, 95.00, 'Ocupata');
```

```
INSERT INTO PLANTE VALUES (seq_plante.NEXTVAL, 'Rosii', 'Solanum lycopersicum', 'Udare regulata,  
multa lumina', 90, 'Vara');
```

```
INSERT INTO PLANTE VALUES (seq_plante.NEXTVAL, 'Castraveti', 'Cucumis sativus', 'Udare moderata, suport pentru crestere', 60, 'Vara');
```

```
INSERT INTO PLANTE VALUES (seq_plante.NEXTVAL, 'Salata', 'Lactuca sativa', 'Udare regulata, umbra partiala', 45, 'Primavara');
```

```
INSERT INTO PLANTE VALUES (seq_plante.NEXTVAL, 'Morcovi', 'Daucus carota', 'Sol afanat, udare moderata', 70, 'Tot anul');
```

```
INSERT INTO PLANTE VALUES (seq_plante.NEXTVAL, 'Ceapa', 'Allium cepa', 'Sol bine drenat, udare redusa', 100, 'Primavara');
```

```
INSERT INTO PLANTE VALUES (seq_plante.NEXTVAL, 'Ardei', 'Capsicum annuum', 'Multa lumina, caldura', 80, 'Vara');
```

```
INSERT INTO ATRIBUIRI_PARCELE VALUES (seq_atribuiri.NEXTVAL, 1, 1, TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('2024-12-31', 'YYYY-MM-DD'));
```

```
INSERT INTO ATRIBUIRI_PARCELE VALUES (seq_atribuiri.NEXTVAL, 2, 2, TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('2024-12-31', 'YYYY-MM-DD'));
```

```
INSERT INTO ATRIBUIRI_PARCELE VALUES (seq_atribuiri.NEXTVAL, 3, 3, TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('2024-12-31', 'YYYY-MM-DD'));
```

```
INSERT INTO ATRIBUIRI_PARCELE VALUES (seq_atribuiri.NEXTVAL, 4, 6, TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('2024-12-31', 'YYYY-MM-DD'));
```

```
INSERT INTO ATRIBUIRI_PARCELE VALUES (seq_atribuiri.NEXTVAL, 5, 1, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-12-31', 'YYYY-MM-DD'));
```

```
INSERT INTO ATRIBUIRI_PARCELE VALUES (seq_atribuiri.NEXTVAL, 6, 2, TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2023-12-31', 'YYYY-MM-DD'));
```

```
INSERT INTO UNELTE VALUES (seq_unelte.NEXTVAL, 'Lopata', 'Buna', TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1, 100.00);
```

```
INSERT INTO UNELTE VALUES (seq_unelte.NEXTVAL, 'Grebla', 'Buna', TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1, 50.00);
```

```
INSERT INTO UNELTE VALUES (seq_unelte.NEXTVAL, 'Foarfeca gradinarit', 'Noua', TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1, 75.00);
```

```
INSERT INTO UNELTE VALUES (seq_unelte.NEXTVAL, 'Stropitoare', 'Buna', TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1, 40.00);
```

```
INSERT INTO UNELTE VALUES (seq_unelte.NEXTVAL, 'Furca', 'Necesita Reparatii', TO_DATE('2023-12-01', 'YYYY-MM-DD'), 0, 90.00);
```

```
INSERT INTO UNELTE VALUES (seq_unelte.NEXTVAL, 'Roaba', 'Buna', TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1, 200.00);
```

```
INSERT INTO SARCINI_INTRETINERE VALUES (seq_sarcini.NEXTVAL, 'Plivit buruieni', 'Medie', 120, 1);
```

```
INSERT INTO SARCINI_INTRETINERE VALUES (seq_sarcini.NEXTVAL, 'Fertilizare sol', 'Mare', 90, 1);
```

```
INSERT INTO SARCINI_INTRETINERE VALUES (seq_sarcini.NEXTVAL, 'Tundere gard viu', 'Mica', 60, 1);
```

```
INSERT INTO SARCINI_INTRETINERE VALUES (seq_sarcini.NEXTVAL, 'Verificare sistem irigatie', 'Urgenta', 45, 0);
```

```
INSERT INTO SARCINI_INTRETINERE VALUES (seq_sarcini.NEXTVAL, 'Curatare alei', 'Medie', 90, 1);
```

```
INSERT INTO SARCINI_INTRETINERE VALUES (seq_sarcini.NEXTVAL, 'Pregatire sol', 'Mare', 180, 1);
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 1, 1, 1, TO_DATE('2024-02-01', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 2, 2, 2, TO_DATE('2024-02-02', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 3, 3, 3, TO_DATE('2024-02-03', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 4, 4, 4, TO_DATE('2024-02-04', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 5, 5, 5, TO_DATE('2024-02-05', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 6, 6, 6, TO_DATE('2024-02-06', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 1, 2, 3, TO_DATE('2024-02-07', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 2, 3, 4, TO_DATE('2024-02-08', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 3, 4, 5, TO_DATE('2024-02-09', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO PROGRAMARI_SARCINI VALUES (seq_programari.NEXTVAL, 4, 5, 6, TO_DATE('2024-02-10', 'YYYY-MM-DD'), 'Programata');
```

```
INSERT INTO RECOLTE VALUES (seq_recolte.NEXTVAL, 1, 1, TO_DATE('2024-09-01', 'YYYY-MM-DD'), 25.5, 'Excelenta');
```

```
INSERT INTO RECOLTE VALUES (seq_recolte.NEXTVAL, 2, 2, TO_DATE('2024-08-15', 'YYYY-MM-DD'), 15.3, 'Buna');
```

```
INSERT INTO RECOLTE VALUES (seq_recolte.NEXTVAL, 3, 3, TO_DATE('2024-05-20', 'YYYY-MM-DD'), 8.7, 'Medie');
```

```
INSERT INTO RECOLTE VALUES (seq_recolte.NEXTVAL, 4, 4, TO_DATE('2024-10-10', 'YYYY-MM-DD'), 20.0, 'Buna');
```

```
INSERT INTO RECOLTE VALUES (seq_recolte.NEXTVAL, 5, 5, TO_DATE('2024-06-01', 'YYYY-MM-DD'), 12.8, 'Excelenta');
```

```
INSERT INTO RECOLTE VALUES (seq_recolte.NEXTVAL, 1, 6, TO_DATE('2024-09-15', 'YYYY-MM-DD'), 18.2, 'Buna');
```

```
INSERT INTO PLANTE_PARCELE VALUES (1, 1, TO_DATE('2024-04-01', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (2, 1, TO_DATE('2024-04-01', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (3, 2, TO_DATE('2024-03-15', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (4, 2, TO_DATE('2024-03-15', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (5, 3, TO_DATE('2024-03-01', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (6, 3, TO_DATE('2024-03-01', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (1, 4, TO_DATE('2024-04-15', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (2, 4, TO_DATE('2024-04-15', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (3, 5, TO_DATE('2024-03-20', 'YYYY-MM-DD'));
```

```
INSERT INTO PLANTE_PARCELE VALUES (4, 6, TO_DATE('2024-03-20', 'YYYY-MM-DD'));
```

```
INSERT INTO UNELTE_SARCINI VALUES (1, 1, 2);
```

```
INSERT INTO UNELTE_SARCINI VALUES (2, 1, 1);
```

```
INSERT INTO UNELTE_SARCINI VALUES (3, 2, 1);
```

```
INSERT INTO UNELTE_SARCINI VALUES (4, 2, 2);
```

```
INSERT INTO UNELTE_SARCINI VALUES (5, 3, 1);
```

```
INSERT INTO UNELTE_SARCINI VALUES (6, 3, 1);
```

```
INSERT INTO UNELTE_SARCINI VALUES (1, 4, 1);
```

```
INSERT INTO UNELTE_SARCINI VALUES (2, 5, 2);
```

```
INSERT INTO UNELTE_SARCINI VALUES (3, 6, 1);
```

```
INSERT INTO UNELTE_SARCINI VALUES (4, 6, 2);
```

```
COMMIT;
```



## 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.

Sa se scrie o procedura care primeste ca parametru un id de parcela si afiseaza membrii activi, plantele cultivate pe parcela specificata si toate cheltuielile estimate pentru sarcini

The screenshot displays an SQL Worksheet with a PL/SQL procedure named `GESTIONARE_DATE` and its execution results.

**Procedure Code:**

```

40 FOR rec IN (
41     SELECT p.num
42     FROM PLANTE p
43     JOIN PLANTE_PARCELE pp ON p.planta_id = pp.planta_id
44     WHERE pp.atribuire_parcela_id = p_parcela_id
45 ) LOOP
46     plante_varray.EXTEND;
47     plante_varray(plante_varray.LAST) := rec.num;
48 END LOOP;
49
50 DBMS_OUTPUT.PUT_LINE('Plante cultivate in parcela specificata:');
51 FOR i IN 1..plante_varray.COUNT LOOP
52     DBMS_OUTPUT.PUT_LINE(' ' || plante_varray(i));
53 END LOOP;
54
55 FOR rec IN (
56     SELECT sarcina_id, durata_estimata * 10 AS cheltuiala_estimativa
57     FROM SARCINI_INTRETINERE
58 ) LOOP
59     cheltuieli_table(rec.sarcina_id) := rec.cheltuiala_estimativa;
60 END LOOP;
61
62 DBMS_OUTPUT.PUT_LINE('Cheltuieli estimate pentru sarcini:');
63 v_sarcina_id := cheltuieli_table.FIRST;
64 WHILE v_sarcina_id IS NOT NULL LOOP
65     DBMS_OUTPUT.PUT_LINE('Sarcina ID ' || v_sarcina_id || ': ' || cheltuieli_table(v_sarcina_id) || ' RON');
66     v_sarcina_id := cheltuieli_table.NEXT(v_sarcina_id);
67 END LOOP;
68 END;
69 /
70 BEGIN
71     gestionare_date(2);
72 END;
73 /

```

**Output Results:**

Membrii activi:  
 Popescu Ion (popescu.ion@email.com)  
 Ionescu Maria (ionescu.maria@email.com)  
 Popa Ana (popa.ana@email.com)  
 Dumitrescu Vasile (dumitrescu.vasile@email.com)  
 Stanescu Elena (stanescu.elena@email.com)  
 Georgescu Andrei (georgescu.andrei@email.com)

Plante cultivate in parcela specificata:  
 - Salata  
 - Morcovi

Cheltuieli estimate pentru sarcini:  
 Sarcina ID 1: 1200 RON  
 Sarcina ID 2: 900 RON  
 Sarcina ID 3: 600 RON  
 Sarcina ID 4: 450 RON  
 Sarcina ID 5: 900 RON  
 Sarcina ID 6: 1800 RON

Script Output x Query Result x  
 Task completed in 0.026 seconds  
 Procedure GESTIONARE\_DATE compiled  
 PL/SQL procedure successfully completed.

```
CREATE OR REPLACE PROCEDURE gestionare_date(  
    p_parcela_id IN NUMBER  
) AS  
    TYPE t_membri IS TABLE OF VARCHAR2(200);  
    membri_table t_membri;  
  
    -- Varray pentru plante  
    TYPE t_plante IS VARRAY(100) OF VARCHAR2(50);  
    plante_varray t_plante;  
  
    TYPE t_cheltuieli IS TABLE OF NUMBER INDEX BY PLS_INTEGER;  
    cheltuieli_table t_cheltuieli;  
  
    v_nume VARCHAR2(50);  
    v_email VARCHAR2(100);  
    v_planta VARCHAR2(50);  
    v_sarcina_id NUMBER;  
    v_cheltuiala NUMBER;  
  
BEGIN  
    membri_table := t_membri();  
    FOR rec IN (  
        SELECT nume || ' ' || prenume AS nume_complet, email  
        FROM MEMBRII  
    ) LOOP  
        membri_table.EXTEND;  
        membri_table(membri_table.LAST) := rec.nume_complet || ' (' || rec.email || ')';  
    END LOOP;
```

```
DBMS_OUTPUT.PUT_LINE('Membrii activi:');
FOR i IN 1..membri_table.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(membri_table(i));
END LOOP;

-- Populare Varray cu plante din parcela specificat?
plante_varray := t_plante();
FOR rec IN (
    SELECT p.nume
    FROM PLANTE p
    JOIN PLANTE_PARCELE pp ON p.planta_id = pp.planta_id
    WHERE pp.atribuire_parcela_id = p_parcela_id
) LOOP
    plante_varray.EXTEND;
    plante_varray(plante_varray.LAST) := rec.nume;
END LOOP;

DBMS_OUTPUT.PUT_LINE('Plante cultivate n parcela specificata:');
FOR i IN 1..plante_varray.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('- ' || plante_varray(i));
END LOOP;

FOR rec IN (
    SELECT sarcina_id, durata_estimata * 10 AS cheltuiala_estimativa
    FROM SARCINI_INTRETINERE
) LOOP
    cheltuieli_table(rec.sarcina_id) := rec.cheltuiala_estimativa;
END LOOP;
```

```
DBMS_OUTPUT.PUT_LINE('Cheltuieli estimate pentru sarcini:');  
v_sarcina_id := cheltuieli_table.FIRST;  
WHILE v_sarcina_id IS NOT NULL LOOP  
    DBMS_OUTPUT.PUT_LINE('Sarcina ID ' || v_sarcina_id || ': ' || cheltuieli_table(v_sarcina_id) ||  
' RON');  
    v_sarcina_id := cheltuieli_table.NEXT(v_sarcina_id);  
END LOOP;  
END;  
/  
BEGIN  
    gestionare_date(2);  
END;  
/
```

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.

Sa se creeze o procedura care sa afiseze pentru fiecare membru al gradinii lista plantelor pe care le cultiva în parcelele atribuite.

The screenshot displays the sgbd SQL editor interface. The main window shows the source code for a PL/SQL procedure named `membrii_Plante`. The code defines two cursors: a simple cursor `c_membrii` for selecting all members, and a parametrized cursor `c_plante` for selecting plants assigned to a specific member. The procedure uses nested loops to iterate over members and their assigned plants, outputting the results using `DBMS_OUTPUT.PUT_LINE`.

```

1 CREATE OR REPLACE PROCEDURE membrii_Plante IS
2
3   --cursor simplu
4   CURSOR c_membrii IS
5     SELECT *
6     FROM membrii;
7
8   --cursor parametrizat
9   CURSOR c_plante(p_membru_id NUMBER) IS
10    SELECT pl.ume
11    FROM plante pl
12    JOIN plante_parcele pp ON pp.planta_id = pl.planta_id
13    JOIN atribuire_parcele ap ON pp.atribuire_parcela_id = ap.atribuire_id
14    WHERE ap.membru_id = p_membru_id;
15
16  BEGIN
17
18    FOR membru IN c_membrii LOOP
19      DBMS_OUTPUT.PUT_LINE(membru.Nume || ' ' || membru.prenume || ' cultiva: ');
20      FOR planta IN c_plante(membru.membru_id) LOOP
21
22        DBMS_OUTPUT.PUT_LINE(planta.ume);
23
24      END LOOP;
25      DBMS_OUTPUT.PUT_LINE(' ');
26    END LOOP;
27
28  END membrii_plante;
29
30
31 /
32
33 BEGIN
34   membrii_plante();
35 END;
36
37

```

The right-hand pane, titled "Dbms Output", shows the execution results of the procedure. It lists the members and the plants they are cultivating:

```

Popescu Ion cultiva:
Rosii
Castraveti

Ionescu Maria cultiva:
Salata
Morcovi

Popa Ana cultiva:
Ceapa
Ardei

Dumitrescu Vasile cultiva:
Rosii
Castraveti

Stanescu Elena cultiva:
Salata

Georgescu Andrei cultiva:
Morcovi

```

The bottom pane, titled "Query Result", shows the status of the procedure execution:

```

Procedure MEMBRII_PLANTE compiled

PL/SQL procedure successfully completed.

```

The task completed in 0.025 seconds.

CREATE OR REPLACE PROCEDURE membrii\_Plante IS

--cursor simplu

CURSOR c\_membrii IS

SELECT \*

FROM membrii;

--cursor parametrizat

CURSOR c\_plante(p\_membru\_id NUMBER) IS

SELECT pl.nume

FROM plante pl

JOIN plante\_parcele pp ON pp.planta\_id = pl.planta\_id

JOIN atribuii\_parcele ap ON pp.atribuire\_parcela\_id = ap.atribuire\_id

WHERE ap.membru\_id = p\_membru\_id;

BEGIN

FOR membru IN c\_membrii LOOP

DBMS\_OUTPUT.PUT\_LINE(membru.Nume || ' ' || membru.prenume || ' cultiva: ');

FOR planta IN c\_plante(membru.membru\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(planta.nume);

END LOOP;

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

END LOOP;

END membrii\_plante;

/

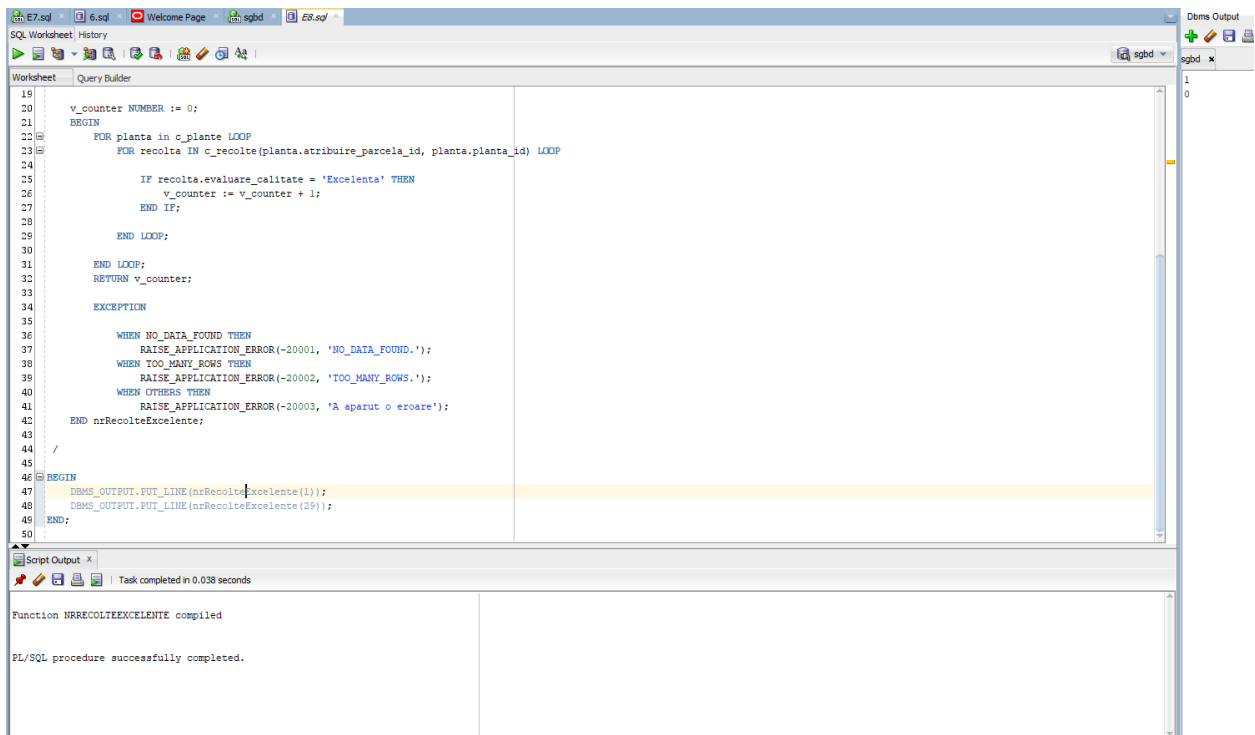
BEGIN

    membrii\_plante();

END;

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.

Scrieti o functie care sa ia ca parametru un id de membru si sa returneze numarul de recolte care au calitatea Excelenta



```

19
20 v_counter NUMBER := 0;
21 BEGIN
22     FOR planta IN c_plante LOOP
23         FOR recolta IN c_recolte(planta.attribuire_parcela_id, planta.planta_id) LOOP
24
25             IF recolta.evaluare_calitate = 'Excelenta' THEN
26                 v_counter := v_counter + 1;
27             END IF;
28
29         END LOOP;
30     END LOOP;
31     RETURN v_counter;
32
33     EXCEPTION
34
35         WHEN NO_DATA_FOUND THEN
36             RAISE_APPLICATION_ERROR(-20001, 'NO_DATA_FOUND. ');
37         WHEN TOO_MANY_ROWS THEN
38             RAISE_APPLICATION_ERROR(-20002, 'TOO_MANY_ROWS. ');
39         WHEN OTHERS THEN
40             RAISE_APPLICATION_ERROR(-20003, 'A aparut o eroare');
41     END nrRecolteExcelente;
42
43 /
44
45
46 BEGIN
47     DBMS_OUTPUT.PUT_LINE(nrRecolteExcelente(1));
48     DBMS_OUTPUT.PUT_LINE(nrRecolteExcelente(25));
49 END;
50

```

Script Output x

Task completed in 0.038 seconds

Function NRRECOLTEEXCELENTE compiled

PL/SQL procedure successfully completed.



```
CREATE OR REPLACE FUNCTION nrRecolteExcelente(p_membru_id NUMBER)
```

```
RETURN NUMBER
```

```
IS
```

```
CURSOR c_recolte(p_atribuire_parcela_id NUMBER, p_id NUMBER)IS
```

```
SELECT *
```

```
FROM recolte
```

```
WHERE recolte.atribuire_parcela_id = p_atribuire_parcela_id and recolte.planta_id = p_id;
```

```
CURSOR c_plante IS
```

```
SELECT pp.atribuire_parcela_id, pl.planta_id
```

```
FROM plante pl
```

```
JOIN plante_parcele pp ON pp.planta_id = pl.planta_id
```

```
JOIN atribuii_parcele ap ON pp.atribuire_parcela_id = ap.atribuire_id
```

```
WHERE ap.membru_id = p_membru_id;
```

```
v_counter NUMBER := 0;
```

```
BEGIN
```

```
FOR planta in c_plante LOOP
```

```
FOR recolta IN c_recolte(planta.atribuire_parcela_id, planta.planta_id) LOOP
```

```
IF recolta.evaluare_calitate = 'Excelenta' THEN
```

```
v_counter := v_counter + 1;
```

```
END IF;
```

```
END LOOP;
```

```
END LOOP;
```

```
RETURN v_counter;
```

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20001, 'NO\_DATA\_FOUND.');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'TOO\_MANY\_ROWS.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20003, 'A aparut o eroare');

END nrRecolteExcelente;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(nrRecolteExcelente(1));

DBMS\_OUTPUT.PUT\_LINE(nrRecolteExcelente(29));

END;

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.

--Scrieti o procedura care primeste ca parametru un id de membru, o locatie de interes si un sezon (optional) Aceasta procedura verifica pentru un anumit membru daca acesta are cel putin o planta cultivata intr-o anumita locatie si daca planta creste intr-un anumit sezon

The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL procedure named `VERIFICARE_MEMBRU_PLANTIE` with the following code:

```

85 BEGIN
86     verificare_membru_plante(1, 'Zona A1', 'Vara');
87 END;
88 /
89 BEGIN
90     verificare_membru_plante(5, 'Zona B1', NULL);
91 END;
92 /
93 BEGIN
94     verificare_membru_plante(3, 'Zona B1', NULL);
95 END;

```

The bottom pane shows the execution results, including the compiled procedure and two error messages:

```

Procedure VERIFICARE_MEMBRU_PLANTIE compiled

PL/SQL procedure successfully completed.

Error starting at line : 99 in command -
BEGIN
    verificare_membru_plante(5, 'Zona B1', NULL);
END;
Error report -
ORA-20004: Membrul introdus nu are parcela respectiva.
ORA-06512: at "DARIUS.VERIFICARE_MEMBRU_PLANTIE", line 81
ORA-06512: at line 2

Error starting at line : 103 in command -
BEGIN
    verificare_membru_plante(3, 'Zona B1', NULL);
END;
Error report -
ORA-20005: Pe parcela respectiva, membrul respectiv nu cultiva plante pentru sezonul respectiv.
ORA-06512: at "DARIUS.VERIFICARE_MEMBRU_PLANTIE", line 83
ORA-06512: at line 2

```

The right pane shows the output of the procedure, displaying the name, prename, plant, and location for each call:

```

Name: Popescu, Prenume: Ion, Planta: Rosii, Locatie: Zona A1
Name: Popescu, Prenume: Ion, Planta: Castraveti, Locatie: Zona A1

```

```
CREATE OR REPLACE PROCEDURE verificare_membru_plante(
```

```
    p_membru_id NUMBER,  
    p_locatie VARCHAR2,  
    p_sezon VARCHAR2 DEFAULT NULL
```

```
)
```

```
IS
```

```
TYPE t_membru IS RECORD(  
    v_nume      VARCHAR2(50),  
    v_prenume   VARCHAR2(50),  
    v_nume_planta VARCHAR2(50),  
    v_locatie_parcela VARCHAR2(100)  
);
```

```
TYPE t_plante IS TABLE OF t_membru;  
membru_plante t_plante := t_plante();
```

```
exceptie_parcela_invalida EXCEPTION;  
exceptie_sezon_invalid EXCEPTION;
```

```
v_count_parcele NUMBER := 0;  
v_count_plante NUMBER := 0;
```

```
CURSOR c_plante IS
```

```
    SELECT m.nume, m.prenume, pl.nume AS nume_planta, p.locatie AS locatie_parcela  
    FROM MEMBRII m  
    JOIN ATRIBUIRI_PARCELE ap ON ap.membru_id = m.membru_id  
    JOIN PARCELE p ON p.parcela_id = ap.parcela_id  
    JOIN PLANTE_PARCELE pp ON pp.atribuire_parcela_id = ap.atribuire_id
```

```
JOIN PLANTE pl ON pl.planta_id = pp.planta_id
WHERE m.membru_id = p_membru_id
      AND p.locatie = p_locatie
      AND (p_sezon IS NULL OR pl.sezon = p_sezon);
BEGIN

SELECT COUNT(*)
      INTO v_count_parcele
FROM MEMBRII m
JOIN ATRIBUIRI_PARCELE ap ON ap.membru_id = m.membru_id
JOIN PARCELE p ON p.parcela_id = ap.parcela_id
WHERE m.membru_id = p_membru_id AND p.locatie = p_locatie;

IF v_count_parcele = 0 THEN
      RAISE exceptie_parcela_invalida;
END IF;

IF p_sezon IS NULL THEN
      SELECT COUNT(*)
            INTO v_count_plante
      FROM MEMBRII m
      JOIN ATRIBUIRI_PARCELE ap ON ap.membru_id = m.membru_id
      JOIN PARCELE p ON p.parcela_id = ap.parcela_id
      JOIN PLANTE_PARCELE pp ON pp.atribuire_parcela_id = ap.atribuire_id
      JOIN PLANTE pl ON pl.planta_id = pp.planta_id
      WHERE m.membru_id = p_membru_id AND p.locatie = p_locatie AND (pl.sezon IS NULL);
ELSE
      SELECT COUNT(*)
```

```
        INTO v_count_plante
        FROM MEMBRII m
        JOIN ATRIBUIRI_PARCELE ap ON ap.membru_id = m.membru_id
        JOIN PARCELE p ON p.parcela_id = ap.parcela_id
        JOIN PLANTE_PARCELE pp ON pp.atribuire_parcela_id = ap.atribuire_id
        JOIN PLANTE pl ON pl.planta_id = pp.planta_id
        WHERE m.membru_id = p_membru_id AND p.locatie = p_locatie AND pl.sezon = p_sezon;
    END IF;

    IF v_count_plante = 0 THEN
        RAISE exceptie_sezon_invalid;
    END IF;

    FOR r_planta IN c_plante LOOP
        DBMS_OUTPUT.PUT_LINE('Nume: ' || r_planta.num_e || ', Prenume: ' || r_planta.prenume ||
            ', Planta: ' || r_planta.num_e_planta || ', Locatie: ' || r_planta.locatie_parcela);
    END LOOP;

    EXCEPTION

        WHEN exceptie_parcela_invalida THEN
            RAISE_APPLICATION_ERROR(-20004, 'Membrul introdus nu are parcela respectiva.');
```

```
        WHEN exceptie_sezon_invalid THEN
            RAISE_APPLICATION_ERROR(-20005, 'Pe parcela respectiva, membrul respectiv nu cultiva plante
        pentru sezonul respectiv.');
```

```
        WHEN NO_DATA_FOUND THEN
```

```
        RAISE_APPLICATION_ERROR(-20001, 'NO_DATA_FOUND.');
```

WHEN TOO\_MANY\_ROWS THEN

```
        RAISE_APPLICATION_ERROR(-20002, 'TOO_MANY_ROWS.');
```

WHEN OTHERS THEN

```
        RAISE_APPLICATION_ERROR(-20003, 'A aparut o eroare');
```

end verificare\_membru\_plante;

/

```
BEGIN
```

verificare\_membru\_plante(1, 'Zona A1', 'Vara');

```
END;
```

/

```
BEGIN
```

verificare\_membru\_plante(5, 'Zona B1', NULL);

```
END;
```

/

```
BEGIN
```

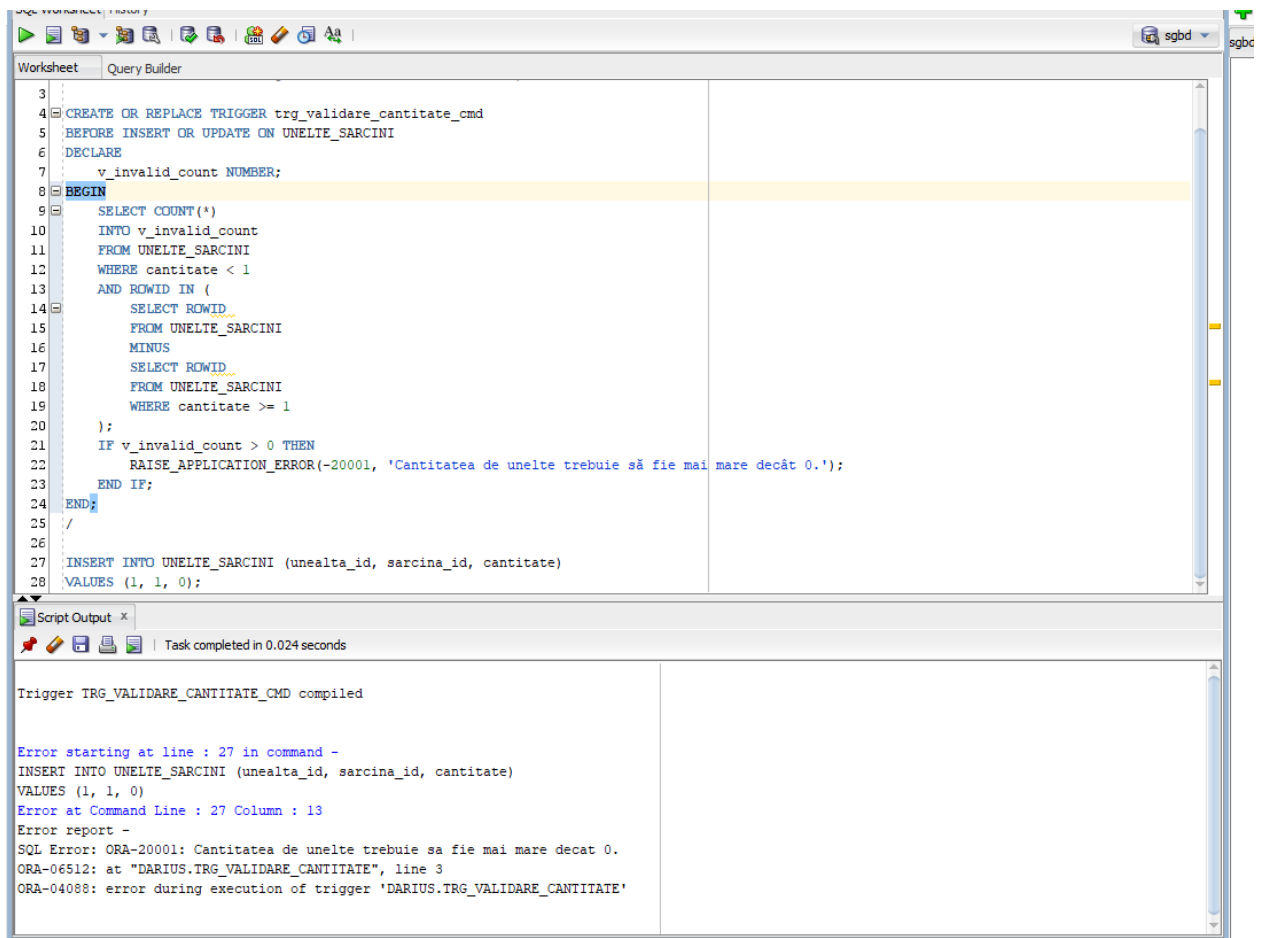
verificare\_membru\_plante(3, 'Zona B1', NULL);

```
END;
```

/

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

Sa se implementeze un trigger LMD la nivel de comanda care verifica daca atunci cand se adauga o unealta noua la o sarcina, cantitatea nu este zero.



The screenshot displays the SQL Developer interface. The top pane, titled 'Query Builder', contains the following SQL code:

```
3
4 CREATE OR REPLACE TRIGGER trg_validare_cantitate_cmd
5 BEFORE INSERT OR UPDATE ON UNELTE_SARCINI
6 DECLARE
7     v_invalid_count NUMBER;
8 BEGIN
9     SELECT COUNT(*)
10    INTO v_invalid_count
11   FROM UNELTE_SARCINI
12  WHERE cantitate < 1
13     AND ROWID IN (
14         SELECT ROWID
15        FROM UNELTE_SARCINI
16       MINUS
17       SELECT ROWID
18        FROM UNELTE_SARCINI
19       WHERE cantitate >= 1
20     );
21     IF v_invalid_count > 0 THEN
22         RAISE_APPLICATION_ERROR(-20001, 'Cantitatea de unelte trebuie să fie mai mare decât 0.');
```

The bottom pane, titled 'Script Output', shows the execution results:

```
Trigger TRG_VALIDARE_CANTITATE_CMD compiled

Error starting at line : 27 in command -
INSERT INTO UNELTE_SARCINI (unealta_id, sarcina_id, cantitate)
VALUES (1, 1, 0)
Error at Command Line : 27 Column : 13
Error report -
SQL Error: ORA-20001: Cantitatea de unelte trebuie sa fie mai mare decat 0.
ORA-06512: at "DARIUS.TRG_VALIDARE_CANTITATE", line 3
ORA-04088: error during execution of trigger 'DARIUS.TRG_VALIDARE_CANTITATE'
```



```
CREATE OR REPLACE TRIGGER trg_validare_cantitate_cmd
BEFORE INSERT OR UPDATE ON UNELTE_SARCINI
DECLARE
    v_invalid_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_invalid_count
    FROM UNELTE_SARCINI
    WHERE cantitate < 1
    AND ROWID IN (
        SELECT ROWID
        FROM UNELTE_SARCINI
        MINUS
        SELECT ROWID
        FROM UNELTE_SARCINI
        WHERE cantitate >= 1
    );
    IF v_invalid_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cantitatea de unelte trebuie s? fie mai mare decît 0. ');
    END IF;
END;
/

INSERT INTO UNELTE_SARCINI (unealta_id, sarcina_id, cantitate)
VALUES (1, 1, 0);
```

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

Definiți un trigger LMD care verifica dacă o unealta pe care dorim să o adăugăm în unelte\_sarcini nu necesită reparații.

The screenshot displays the Oracle SQL Developer interface. The top pane, titled 'Query Builder', contains a SQL script. The script defines a trigger named 'trg\_validare\_calitate\_unealta' that fires before insert or update on the 'UNELTE\_SARCINI' table. The trigger logic declares two variables, 'v\_id\_unealta' and 'v\_calitate', and performs a SELECT query to check the 'stare' (status) of the equipment in the 'unelte' table. If the status is 'Necesita Reparatii', it raises an application error with message number -20001 and the text 'Unealta este stricata.'. The script concludes with an INSERT statement into 'UNELTE\_SARCINI' with values (5, 10, 1).

```
1 --Definiti un trigger LMD care verifica daca
2 --o unealta pe care dorim sa o adaugam in unelte_sarcini
3 --nu necesita reparatii
4
5
6 CREATE OR REPLACE TRIGGER trg_validare_calitate_unealta
7 BEFORE INSERT OR UPDATE ON UNELTE_SARCINI
8 FOR EACH ROW
9 DECLARE
10     v_id_unealta NUMBER := :NEW.unealta_id;
11     v_calitate VARCHAR2(20) ;
12 BEGIN
13
14     SELECT u.stare
15         INTO v_calitate
16         FROM unelte u
17         WHERE u.unealta_id = v_id_unealta;
18
19     IF v_calitate = 'Necesita Reparatii' THEN
20         RAISE_APPLICATION_ERROR(-20001, 'Unealta este stricata.');
```

The bottom pane, titled 'Script Output', shows the execution results. It indicates that the task completed in 0.043 seconds. However, an error occurred starting at line 24 in the command. The error message is: 'Error at Command Line : 24 Column : 13 Error report - SQL Error: ORA-20001: Unealta este stricata. ORA-06512: at "DARIUS.TRG\_VALIDARE\_CALITATE\_UNEALTA", line 12 ORA-04088: error during execution of trigger 'DARIUS.TRG\_VALIDARE\_CALITATE\_UNEALTA''.

```
21 END IF;
22 END;
23 /
24 INSERT INTO UNELTE_SARCINI (unealta_id, sarcina_id, cantitate)
25 VALUES (5, 10, 1);
```

Script Output x  
Task completed in 0.043 seconds  
Error starting at line : 24 in command -  
INSERT INTO UNELTE\_SARCINI (unealta\_id, sarcina\_id, cantitate)  
VALUES (5, 10, 1)  
Error at Command Line : 24 Column : 13  
Error report -  
SQL Error: ORA-20001: Unealta este stricata.  
ORA-06512: at "DARIUS.TRG\_VALIDARE\_CALITATE\_UNEALTA", line 12  
ORA-04088: error during execution of trigger 'DARIUS.TRG\_VALIDARE\_CALITATE\_UNEALTA'

```
CREATE OR REPLACE TRIGGER trg_validare_calitate_unealta
BEFORE INSERT OR UPDATE ON UNELTE_SARCINI
FOR EACH ROW
DECLARE
    v_id_unealta NUMBER := :NEW.unealta_id;
    v_calitate VARCHAR2(20) ;
BEGIN

    SELECT u.stare
    INTO v_calitate
    FROM unelte u
    WHERE u.unealta_id = v_id_unealta;

    IF v_calitate = 'Necesita Reparatii' THEN
        RAISE_APPLICATION_ERROR(-20001, 'Unealta este stricata.');
```

END IF;

END;

/

INSERT INTO UNELTE\_SARCINI (unealta\_id, sarcina\_id, cantitate)

VALUES (5, 10, 1);

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

Definiți un trigger LDD care să ajute la crearea unui registru cu creeri/editări/stergeri de tabele.

```

1 CREATE TABLE Schema_Change_Logs (
2     Event_Type VARCHAR2(50),
3     Object_Type VARCHAR2(50),
4     Object_Name VARCHAR2(255),
5     Event_Time TIMESTAMP
6 );
7 /
8 CREATE OR REPLACE TRIGGER Schema_Change_Log_Trigger
9 AFTER CREATE OR ALTER OR DROP ON SCHEMA
10 DECLARE
11 BEGIN
12     INSERT INTO Schema_Change_Logs (Event_Type, Object_Type, Object_Name, Event_Time)
13     VALUES (ORA_SYSEVENT, ORA_DICT_OBJ_TYPE, ORA_DICT_OBJ_NAME, SYSDATE);
14 END;
15 /
16 CREATE TABLE test_trigger(test_coloana NUMBER(1));
17
18 SELECT * FROM Schema_Change_Logs;

```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.007 seconds

	EVENT_TYPE	OBJECT_TYPE	OBJECT_NAME	EVENT_TIME
1	CREATE	TABLE	TEST_TRIGGER	09-JAN-25 01.27.31.000000000 PM

```
CREATE TABLE Schema_Change_Logs (  
    Event_Type VARCHAR2(50),  
    Object_Type VARCHAR2(50),  
    Object_Name VARCHAR2(255),  
    Event_Time TIMESTAMP  
);  
  
/  
  
CREATE OR REPLACE TRIGGER Schema_Change_Log_Trigger  
AFTER CREATE OR ALTER OR DROP ON SCHEMA  
DECLARE  
BEGIN  
    INSERT INTO Schema_Change_Logs (Event_Type, Object_Type, Object_Name, Event_Time)  
    VALUES (ORA_SYSEVENT, ORA_DICT_OBJ_TYPE, ORA_DICT_OBJ_NAME, SYSDATE);  
END;  
  
/  
  
CREATE TABLE test_trigger(test_coloana NUMBER(1));  
  
SELECT * FROM Schema_Change_Logs;
```