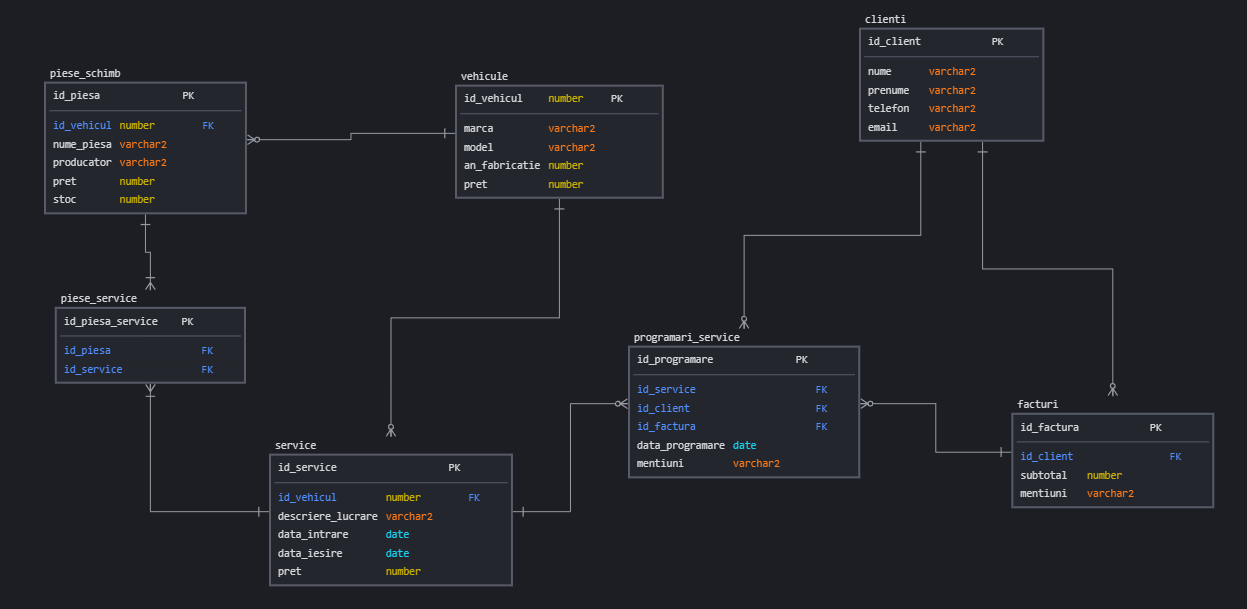
Proiect SGBD – gestiunea unui dealer auto

# Descrierea problemei economice

Baza de date prezentata abordeaza problema gestionarii eficiente a unui service auto, concentrandu-se pe managementul pieselor de schimb, programarilor, clientilor si facturilor. Structura bazei de date include tabele pentru vehicule, piese de schimb, servicii, clienti, programari si facturi, interconectate prin chei primare si straine. Aceasta permite urmarirea stocurilor de piese, detaliilor de service pentru fiecare vehicul, gestionarea programarilor clientilor si generarea facturilor corespunzatoare. Prin centralizarea acestor informatii, baza de date faciliteaza accesul rapid la date esentiale, reduce erorile administrative si imbunatateste experienta clientului printr-o organizare eficienta si precisa a activitatilor din service-ul auto.

# Schema relationala



# Script de creare a tabelelor

CREATE TABLE dealer\_clienti (

id\_client NUMBER PRIMARY KEY,

nume VARCHAR2(50),

prenume VARCHAR2(50),

telefon VARCHAR2(15),

email VARCHAR2(50)

);

CREATE TABLE dealer\_vehicule (

id\_vehicul NUMBER PRIMARY KEY,

marca VARCHAR2(50),

model VARCHAR2(50),

an\_fabricatie NUMBER,

pret NUMBER

);

CREATE TABLE dealer\_service (

id\_service NUMBER PRIMARY KEY,

id\_vehicul NUMBER,

descriere\_lucrare VARCHAR2(200),

data\_intrare DATE,

data\_iesire DATE,

pret NUMBER,

CONSTRAINT fk\_service\_vehicul FOREIGN KEY (id\_vehicul) REFERENCES dealer\_vehicule(id\_vehicul)

);

CREATE TABLE dealer\_piese\_schimb (

id\_piesa NUMBER PRIMARY KEY,

id\_vehicul NUMBER,

nume\_piesa VARCHAR2(50),

producator VARCHAR2(50),

pret NUMBER,

stoc NUMBER,

CONSTRAINT fk\_piese\_vehicul FOREIGN KEY (id\_vehicul) REFERENCES dealer\_vehicule(id\_vehicul)

);

CREATE TABLE dealer\_piese\_service (

id\_piesa\_service NUMBER PRIMARY KEY,

id\_piesa NUMBER,

id\_service NUMBER,

CONSTRAINT fk\_piese\_service\_piesa FOREIGN KEY (id\_piesa) REFERENCES dealer\_piese\_schimb(id\_piesa),

CONSTRAINT fk\_piese\_service\_service FOREIGN KEY (id\_service) REFERENCES dealer\_service(id\_service)

);

CREATE TABLE dealer\_facturi (

id\_factura NUMBER PRIMARY KEY,

id\_client NUMBER,

subtotal NUMBER,

mentiuni VARCHAR2(200),

CONSTRAINT fk\_facturi\_clienti FOREIGN KEY (id\_client) REFERENCES dealer\_clienti(id\_client)

);

CREATE TABLE dealer\_programari\_service (

id\_programmare NUMBER PRIMARY KEY,

id\_service NUMBER,

id\_client NUMBER,

id\_factura NUMBER,

data\_programare DATE,

mentiuni VARCHAR2(200),

CONSTRAINT fk\_programari\_service FOREIGN KEY (id\_service) REFERENCES dealer\_service(id\_service),

CONSTRAINT fk\_programari\_client FOREIGN KEY (id\_client) REFERENCES dealer\_clienti(id\_client),

CONSTRAINT fk\_programari\_factura FOREIGN KEY (id\_factura) REFERENCES dealer\_facturi(id\_factura)

);

# Comenzi de populare a tabelelor

INSERT INTO dealer\_clienti (id\_client, nume, prenume, telefon, email) VALUES (1, 'Popescu', 'Ion', '0720123456', 'ion.popescu@example.com');

INSERT INTO dealer\_clienti (id\_client, nume, prenume, telefon, email) VALUES (2, 'Ionescu', 'Maria', '0720987654', 'maria.ionescu@example.com');

INSERT INTO dealer\_clienti (id\_client, nume, prenume, telefon, email) VALUES (3, 'Georgescu', 'Andrei', '0720345678', 'andrei.georgescu@example.com');

INSERT INTO dealer\_clienti (id\_client, nume, prenume, telefon, email) VALUES (4, 'Dumitrescu', 'Elena', '0720765432', 'elena.dumitrescu@example.com');

INSERT INTO dealer\_clienti (id\_client, nume, prenume, telefon, email) VALUES (5, 'Marinescu', 'Alex', '0720987123', 'alex.marinescu@example.com');

INSERT INTO dealer\_vehicule (id\_vehicul, marca, model, an\_fabricatie, pret) VALUES (1, 'Dacia', 'Duster', 2020, 15000);

INSERT INTO dealer\_vehicule (id\_vehicul, marca, model, an\_fabricatie, pret) VALUES (2, 'Renault', 'Clio', 2019, 12000);

INSERT INTO dealer\_vehicule (id\_vehicul, marca, model, an\_fabricatie, pret) VALUES (3, 'Ford', 'Focus', 2018, 13000);

INSERT INTO dealer\_vehicule (id\_vehicul, marca, model, an\_fabricatie, pret) VALUES (4, 'Volkswagen', 'Golf', 2021, 20000);

INSERT INTO dealer\_vehicule (id\_vehicul, marca, model, an\_fabricatie, pret) VALUES (5, 'Toyota', 'Corolla', 2017, 14000);

INSERT INTO dealer\_service (id\_service, id\_vehicul, descriere\_lucrare, data\_intrare, data\_iesire, pret) VALUES (1, 1, 'Schimb ulei', TO\_DATE('2024-05-01', 'YYYY-MM-DD'), TO\_DATE('2024-05-02', 'YYYY-MM-DD'), 100);

INSERT INTO dealer\_service (id\_service, id\_vehicul, descriere\_lucrare, data\_intrare, data\_iesire, pret) VALUES (2, 2, 'Revizie tehnica', TO\_DATE('2024-05-03', 'YYYY-MM-DD'), TO\_DATE('2024-05-04', 'YYYY-MM-DD'), 200);

INSERT INTO dealer\_service (id\_service, id\_vehicul, descriere\_lucrare, data\_intrare, data\_iesire, pret) VALUES (3, 3, 'Inlocuire placute frana', TO\_DATE('2024-05-05', 'YYYY-MM-DD'), TO\_DATE('2024-05-06', 'YYYY-MM-DD'), 150);

INSERT INTO dealer\_service (id\_service, id\_vehicul, descriere\_lucrare, data\_intrare, data\_iesire, pret) VALUES (4, 4, 'Schimb anvelope', TO\_DATE('2024-05-07', 'YYYY-MM-DD'), TO\_DATE('2024-05-08', 'YYYY-MM-DD'), 300);

INSERT INTO dealer\_service (id\_service, id\_vehicul, descriere\_lucrare, data\_intrare, data\_iesire, pret) VALUES (5, 5, 'Verificare generala', TO\_DATE('2024-05-09', 'YYYY-MM-DD'), TO\_DATE('2024-05-10', 'YYYY-MM-DD'), 250);

INSERT INTO dealer\_piese\_schimb (id\_piesa, id\_vehicul, nume\_piesa, producator, pret, stoc) VALUES (1, 1, 'Filtru ulei', 'Bosch', 50, 10);

INSERT INTO dealer\_piese\_schimb (id\_piesa, id\_vehicul, nume\_piesa, producator, pret, stoc) VALUES (2, 2, 'Bujie', 'NGK', 30, 20);

INSERT INTO dealer\_piese\_schimb (id\_piesa, id\_vehicul, nume\_piesa, producator, pret, stoc) VALUES (3, 3, 'Placute frana', 'Brembo', 80, 15);

INSERT INTO dealer\_piese\_schimb (id\_piesa, id\_vehicul, nume\_piesa, producator, pret, stoc) VALUES (4, 4, 'Anvelopa', 'Continental', 100, 25);

INSERT INTO dealer\_piese\_schimb (id\_piesa, id\_vehicul, nume\_piesa, producator, pret, stoc) VALUES (5, 5, 'Baterie auto', 'Varta', 150, 5);

INSERT INTO dealer\_piese\_service (id\_piesa\_service, id\_piesa, id\_service) VALUES (1, 1, 1);

INSERT INTO dealer\_piese\_service (id\_piesa\_service, id\_piesa, id\_service) VALUES (2, 2, 2);

INSERT INTO dealer\_piese\_service (id\_piesa\_service, id\_piesa, id\_service) VALUES (3, 3, 3);

INSERT INTO dealer\_piese\_service (id\_piesa\_service, id\_piesa, id\_service) VALUES (4, 4, 4);

INSERT INTO dealer\_piese\_service (id\_piesa\_service, id\_piesa, id\_service) VALUES (5, 5, 5);

INSERT INTO dealer\_facturi (id\_factura, id\_client, subtotal, mentiuni) VALUES (1, 1, 150, 'Plata cash');

INSERT INTO dealer\_facturi (id\_factura, id\_client, subtotal, mentiuni) VALUES (2, 2, 230, 'Plata card');

INSERT INTO dealer\_facturi (id\_factura, id\_client, subtotal, mentiuni) VALUES (3, 3, 180, 'Plata cash');

INSERT INTO dealer\_facturi (id\_factura, id\_client, subtotal, mentiuni) VALUES (4, 4, 320, 'Plata card');

INSERT INTO dealer\_facturi (id\_factura, id\_client, subtotal, mentiuni) VALUES (5, 5, 275, 'Plata cash');

INSERT INTO dealer\_programari\_service (id\_programmare, id\_service, id\_client, id\_factura, data\_programare, mentiuni) VALUES (1, 1, 1, 1, TO\_DATE('2024-04-30', 'YYYY-MM-DD'), 'Programare online');

INSERT INTO dealer\_programari\_service (id\_programmare, id\_service, id\_client, id\_factura, data\_programare, mentiuni) VALUES (2, 2, 2, 2, TO\_DATE('2024-05-02', 'YYYY-MM-DD'), 'Programare telefonica');

INSERT INTO dealer\_programari\_service (id\_programmare, id\_service, id\_client, id\_factura, data\_programare, mentiuni) VALUES (3, 3, 3, 3, TO\_DATE('2024-05-04', 'YYYY-MM-DD'), 'Programare online');

INSERT INTO dealer\_programari\_service (id\_programmare, id\_service, id\_client, id\_factura, data\_programare, mentiuni) VALUES (4, 4, 4, 4, TO\_DATE('2024-05-06', 'YYYY-MM-DD'), 'Programare telefonica');

INSERT INTO dealer\_programari\_service (id\_programmare, id\_service, id\_client, id\_factura, data\_programare, mentiuni) VALUES (5, 5, 5, 5, TO\_DATE('2024-05-08', 'YYYY-MM-DD'), 'Programare online');

# Structuri de control, cursori impliciti/expliciti

1. Verificam daca exista client in baza de date.

BEGIN

DECLARE

v\_num\_clienti NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_num\_clienti FROM dealer\_clienti;

IF v\_num\_clienti > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Exista ' || v\_num\_clienti || ' clienti in baza de date.');

ELSE

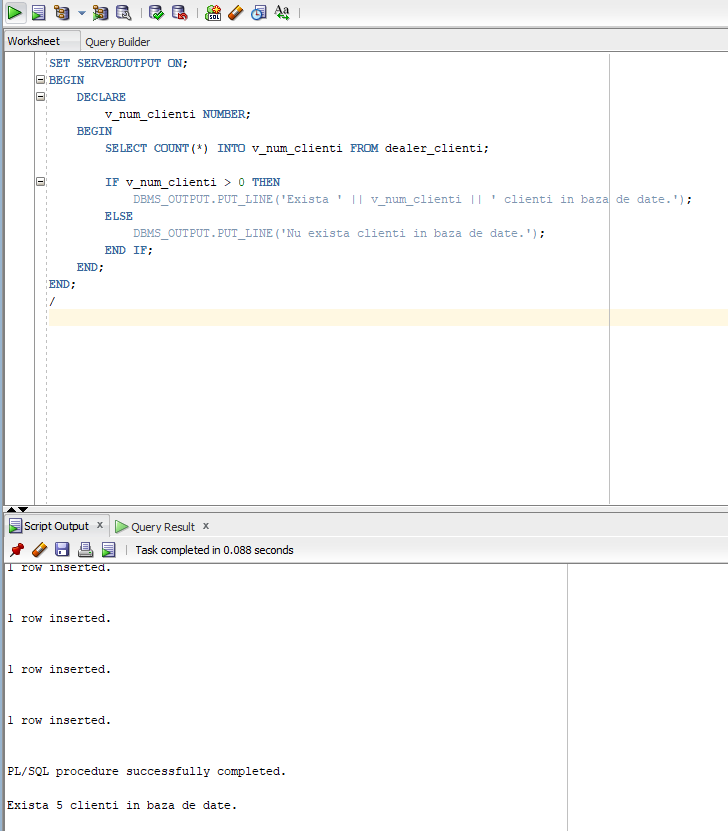
DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti in baza de date.');

END IF;

END;

END;

/



2. Afisarea clientilor care au cel putin o factura in BD printr-un cursor explicit

DECLARE

CURSOR client\_cursor IS

SELECT c.nume, c.prenume

FROM dealer\_clienti c

JOIN dealer\_facturi f ON c.id\_client = f.id\_client;

v\_nume dealer\_clienti.nume%TYPE;

v\_prenume dealer\_clienti.prenume%TYPE;

BEGIN

OPEN client\_cursor;

LOOP

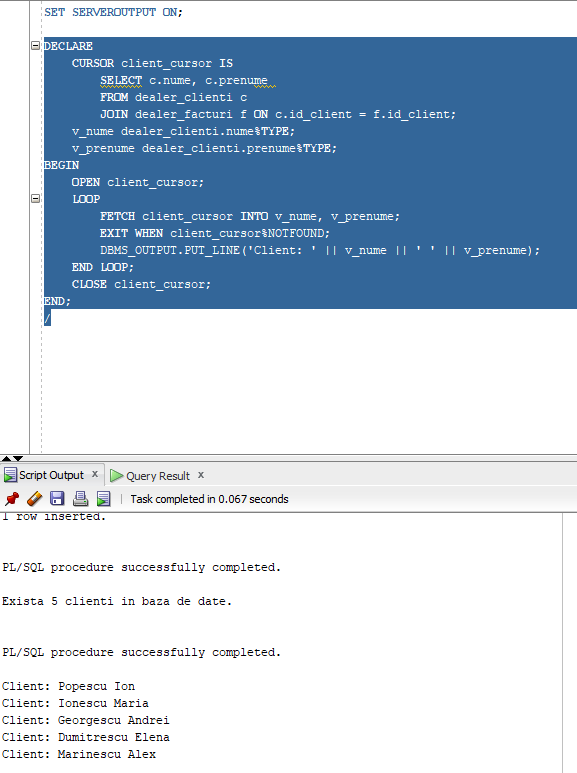
FETCH client\_cursor INTO v\_nume, v\_prenume;

EXIT WHEN client\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Client: ' || v\_nume || ' ' || v\_prenume);

END LOOP;

CLOSE client\_cursor;

END;

/

3. Actualizarea pretului vehiculelor mai vechi de un anumit an. Verificarea nr. de modificari prin cursor implicit

BEGIN

DECLARE

v\_an\_nou NUMBER := 2018;

v\_pret\_nou NUMBER := 10000;

BEGIN

UPDATE dealer\_vehicule

SET pret = v\_pret\_nou

WHERE an\_fabricatie < v\_an\_nou;

IF SQL%ROWCOUNT > 0 THEN

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' vehicule au fost actualizate.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu s-au efectuat modificari.');

END IF;

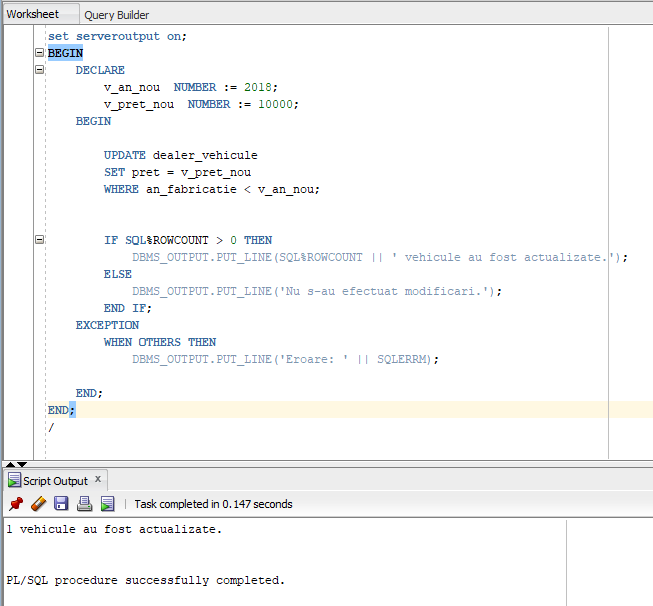
EXCEPTION

WHEN OTHERS THEN

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

END;

END;

/

4. Stergerea clientilor fara facturi folosind cursori explicit/implicit si exceptii aferente.

BEGIN

DECLARE

CURSOR clienti\_cursor IS

SELECT c.id\_client

FROM dealer\_clienti c

WHERE NOT EXISTS (SELECT 1 FROM dealer\_facturi f WHERE f.id\_client = c.id\_client);

v\_id\_client dealer\_clienti.id\_client%TYPE;

v\_exista\_clienti BOOLEAN := FALSE;

BEGIN

OPEN clienti\_cursor;

LOOP

FETCH clienti\_cursor INTO v\_id\_client;

EXIT WHEN clienti\_cursor%NOTFOUND;

v\_exista\_clienti := TRUE;

DELETE FROM dealer\_clienti

WHERE id\_client = v\_id\_client;

IF SQL%ROWCOUNT > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu ID-ul ' || v\_id\_client || ' a fost sters.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Clientul cu ID-ul ' || v\_id\_client || ' nu exista.');

END IF;

END LOOP;

CLOSE clienti\_cursor;

IF NOT v\_exista\_clienti THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost gasiti clienti fara facturi.');

END IF;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

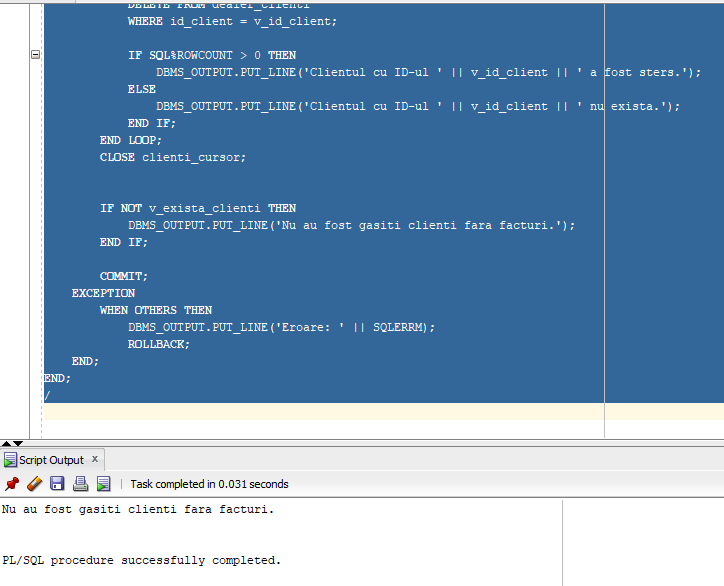
DBMS\_OUTPUT.PUT\_LINE('Eroare: ' || SQLERRM);

ROLLBACK;

END;

END;

/



5. Listarea clientilor si vehiculelor asociate, inclusiv pretul total al pieselor schimbate.

SET SERVEROUTPUT ON;

DECLARE

CURSOR clienti\_vehicule\_cursor IS

SELECT c.id\_client, c.nume, c.prenume, v.id\_vehicul, v.marca, v.model

FROM dealer\_clienti c

JOIN dealer\_programari\_service ps ON c.id\_client = ps.id\_client

JOIN dealer\_service s ON ps.id\_service = s.id\_service

JOIN dealer\_vehicule v ON s.id\_vehicul = v.id\_vehicul;

clienti\_vehicule\_rec clienti\_vehicule\_cursor%ROWTYPE;

CURSOR piese\_cursor (v\_id\_vehicul NUMBER) IS

SELECT SUM(p.pret) AS total\_pret

FROM dealer\_piese\_schimb p

WHERE p.id\_vehicul = v\_id\_vehicul;

piese\_rec piese\_cursor%ROWTYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Lista clienților și vehiculelor asociate:');

OPEN clienti\_vehicule\_cursor;

LOOP

FETCH clienti\_vehicule\_cursor INTO clienti\_vehicule\_rec;

EXIT WHEN clienti\_vehicule\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Client: ' || clienti\_vehicule\_rec.nume || ' ' || clienti\_vehicule\_rec.prenume ||

', Vehicul: ' || clienti\_vehicule\_rec.marca || ' ' || clienti\_vehicule\_rec.model);

OPEN piese\_cursor(clienti\_vehicule\_rec.id\_vehicul);

FETCH piese\_cursor INTO piese\_rec;

IF piese\_cursor%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(' Pret total piese schimbate: ' || NVL(piese\_rec.total\_pret, 0));

ELSE

DBMS\_OUTPUT.PUT\_LINE(' Nu există piese schimbate pentru acest vehicul.');

END IF;

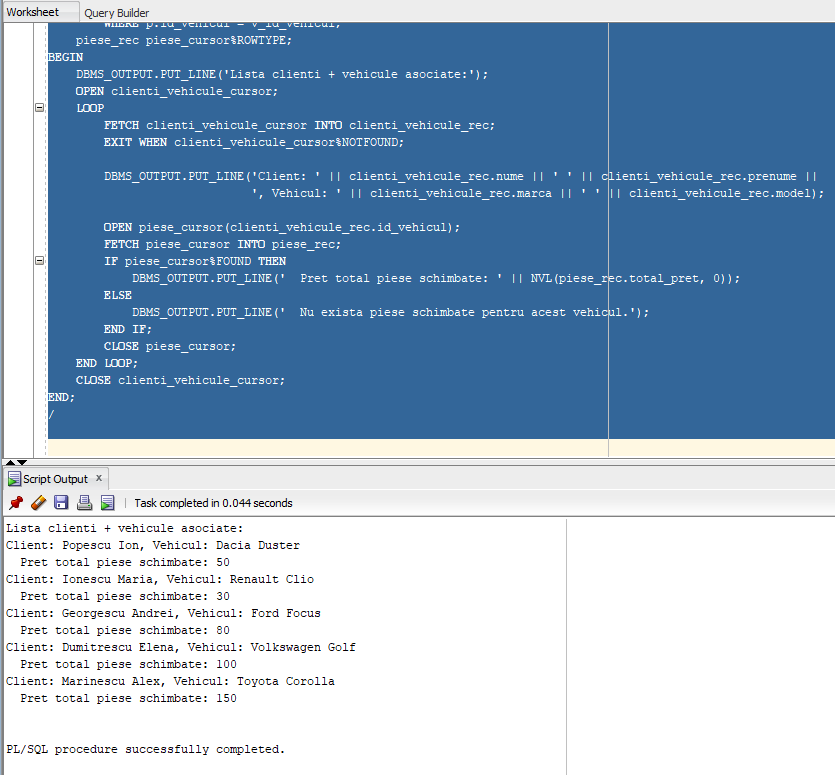
CLOSE piese\_cursor;

END LOOP;

CLOSE clienti\_vehicule\_cursor;

END;

/



# Exceptii implicite/explicite. Tratarea exceptiilor.

1. Verificarea validitatii datei de intrare a unui vehicul in service printr-o exceptie definita de utilizator.

SET SERVEROUTPUT ON;

DECLARE

e\_date\_exception EXCEPTION;

v\_data\_iesire DATE;

BEGIN

SELECT data\_iesire INTO v\_data\_iesire

FROM dealer\_service

WHERE id\_service = 1;

IF v\_data\_iesire > SYSDATE THEN

RAISE e\_date\_exception;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Data de iesire este valida: ' || TO\_CHAR(v\_data\_iesire, 'YYYY-MM-DD'));

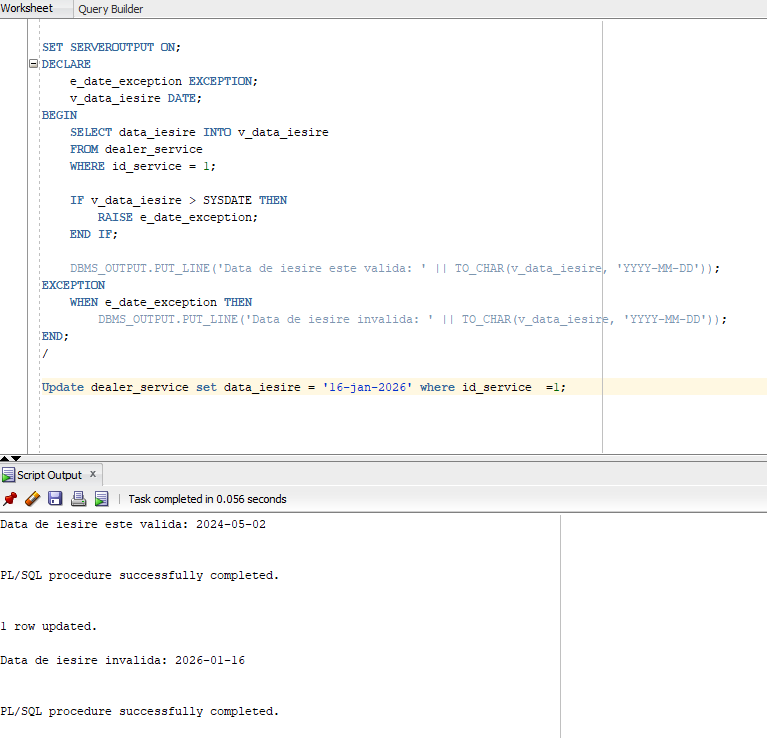
EXCEPTION

WHEN e\_date\_exception THEN

DBMS\_OUTPUT.PUT\_LINE('Data de iesire invalida: ' || TO\_CHAR(v\_data\_iesire, 'YYYY-MM-DD'));

END;

/



1. Verificarea totalului pieselor schimbate pentru un vehicul. Tratarea exceptiilor.

SET SERVEROUTPUT ON;

DECLARE

v\_id\_vehicul NUMBER := 6;

v\_total\_pret NUMBER;

e\_fara\_piese EXCEPTION;

BEGIN

SELECT SUM(p.pret) INTO v\_total\_pret

FROM dealer\_piese\_schimb p

WHERE p.id\_vehicul = v\_id\_vehicul;

IF v\_total\_pret IS NULL THEN

RAISE e\_fara\_piese;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Pretul total al pieselor schimbate pentru vehiculul cu ID ' || v\_id\_vehicul || ' este: ' || v\_total\_pret);

EXCEPTION

WHEN e\_fara\_piese THEN

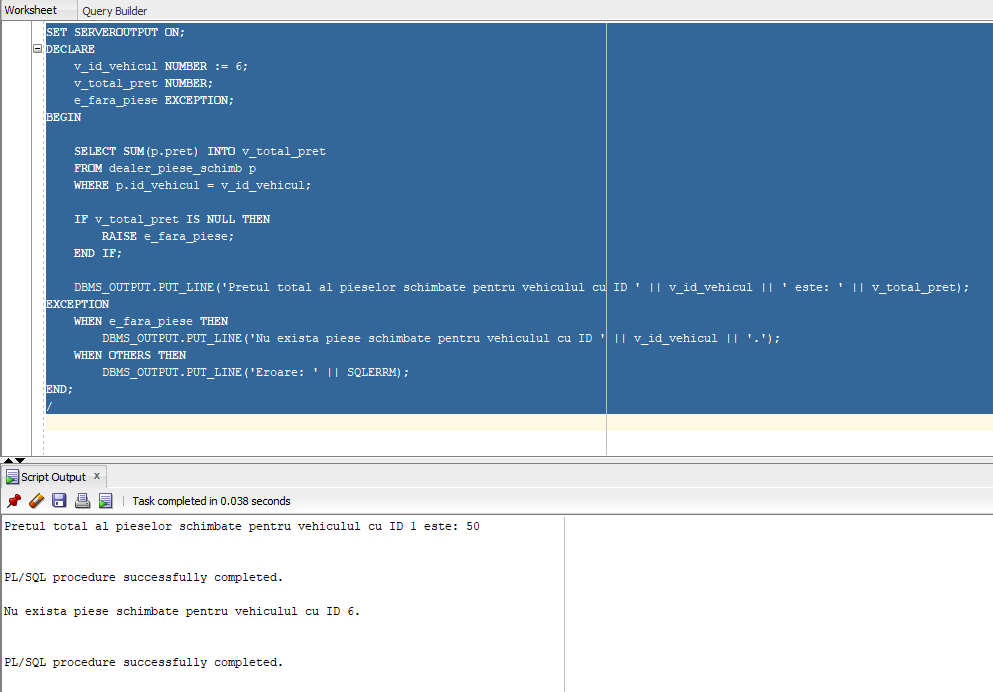
DBMS\_OUTPUT.PUT\_LINE('Nu exista piese schimbate pentru vehiculul cu ID ' || v\_id\_vehicul || '.');

WHEN OTHERS THEN

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

END;

/



1. Crearea unui raport al programarilor si tratarea exceptiilor aferente:

SET SERVEROUTPUT ON;

DECLARE

CURSOR clienti\_cursor IS

SELECT c.id\_client, c.nume, COUNT(ps.id\_programmare) AS numar\_programari, SUM(f.subtotal) AS suma\_facturi

FROM dealer\_clienti c

LEFT JOIN dealer\_programari\_service ps ON c.id\_client = ps.id\_client

LEFT JOIN dealer\_facturi f ON ps.id\_factura = f.id\_factura

GROUP BY c.id\_client, c.nume;

clienti\_rec clienti\_cursor%ROWTYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Raport programari:');

OPEN clienti\_cursor;

LOOP

FETCH clienti\_cursor INTO clienti\_rec;

EXIT WHEN clienti\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Clientul ' || clienti\_rec.nume || ' are ' || clienti\_rec.numar\_programari || ' programari si suma totala a facturilor este: ' || clienti\_rec.suma\_facturi);

END LOOP;

CLOSE clienti\_cursor;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista date disponibile pentru generarea raportului.');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Prea multe randuri gasite');

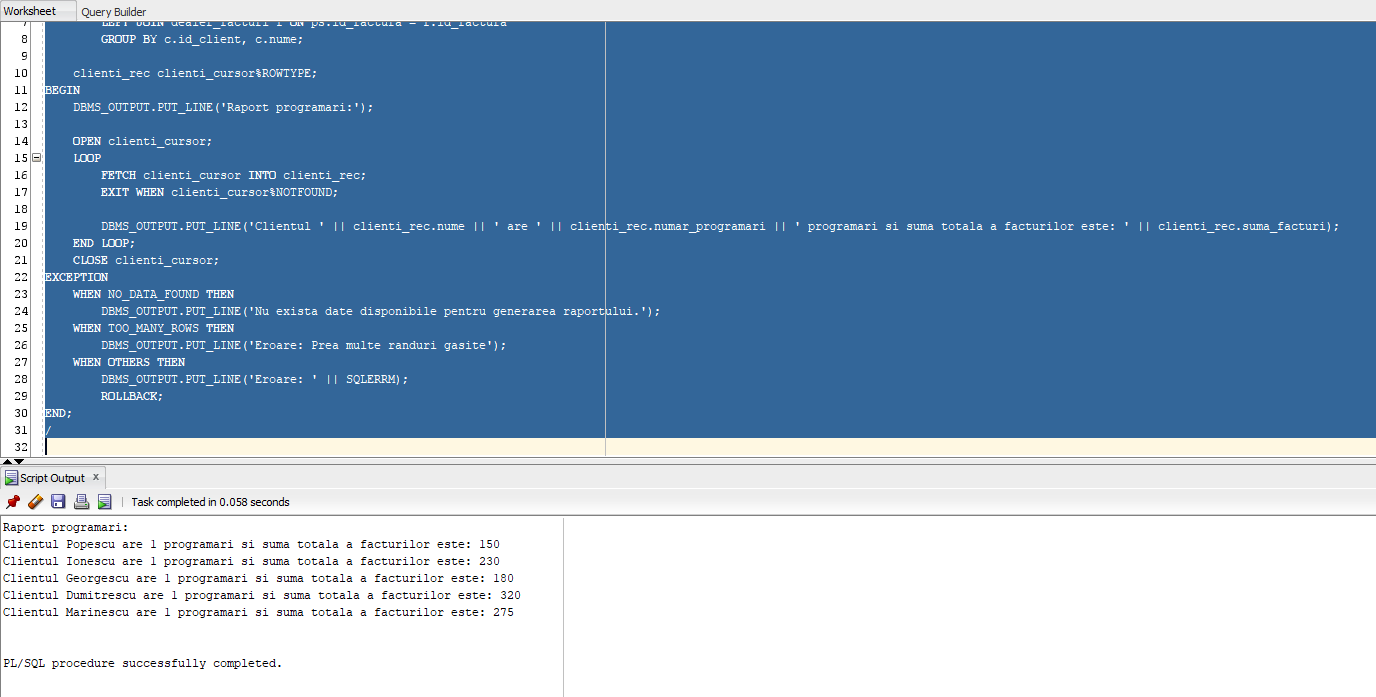
WHEN OTHERS THEN

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

ROLLBACK;

END;

/



1. Verificarea stocului de piese si compararea acestuia cu o valoare luata de la tastatura printr-o variabila de substitutie. Declararea si tratarea exceptiilor.

DECLARE

v\_id\_vehicul dealer\_vehicule.id\_vehicul%TYPE := 1;

v\_id\_piesa dealer\_piese\_schimb.id\_piesa%TYPE;

v\_stoc\_piesa dealer\_piese\_schimb.stoc%TYPE;

v\_limita\_piese NUMBER := &a;

CURSOR c\_piese IS

SELECT id\_piesa, stoc

FROM dealer\_piese\_schimb

WHERE id\_vehicul = v\_id\_vehicul;

e\_stoc\_insuficient EXCEPTION;

BEGIN

OPEN c\_piese;

LOOP

FETCH c\_piese INTO v\_id\_piesa, v\_stoc\_piesa;

EXIT WHEN c\_piese%NOTFOUND;

IF v\_stoc\_piesa < v\_limita\_piese THEN

RAISE e\_stoc\_insuficient;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Stoc suficient pentru piesa cu ID-ul ' || v\_id\_piesa);

END IF;

END LOOP;

CLOSE c\_piese;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista piese de schimb!' || SQLERRM);

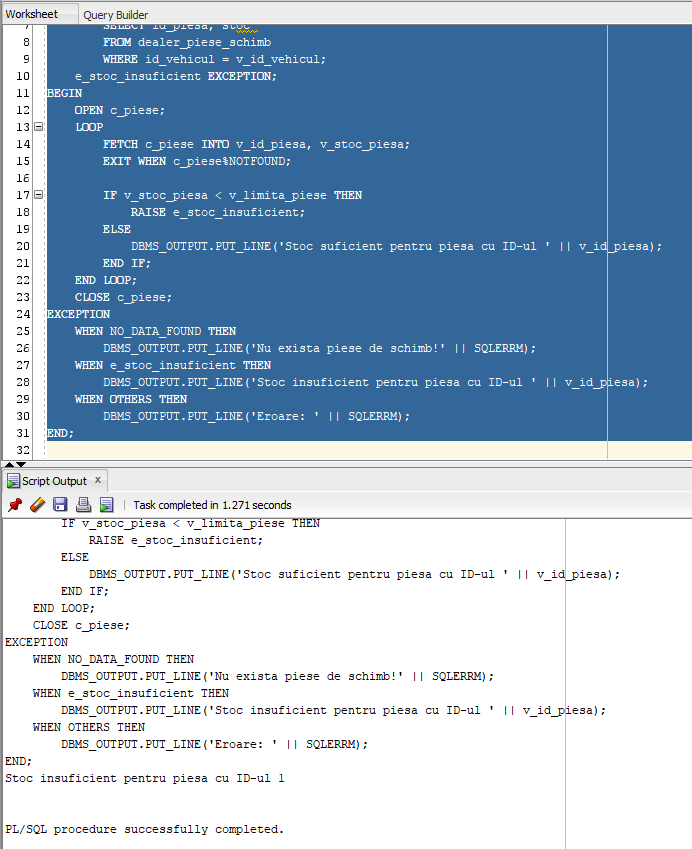
WHEN e\_stoc\_insuficient THEN

DBMS\_OUTPUT.PUT\_LINE('Stoc insuficient pentru piesa cu ID-ul ' || v\_id\_piesa);

WHEN OTHERS THEN

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

END;



# Proceduri si Functii. Pachete de subprograme.

Am creat pachetul “MY\_PROC\_PACK” care contine 4 proceduri.

Antet:

CREATE OR REPLACE PACKAGE MY\_PROC\_PACK IS

PROCEDURE ADAUGARE\_STOC\_PIESA( p\_id IN NUMBER, p\_cantitate IN NUMBER);

PROCEDURE VALOARE\_INVENTAR\_MARCA(p\_marca IN VARCHAR2,p\_valoare OUT NUMBER);

PROCEDURE INSERARE\_CLIENT(p\_id IN NUMBER,p\_nume IN VARCHAR2, p\_prenume IN VARCHAR2, p\_telefon IN VARCHAR2, p\_email IN VARCHAR2);

PROCEDURE STERGERE\_VEHICUL(p\_id IN NUMBER);

END;

Body:

SET SERVEROUTPUT ON;

CREATE OR REPLACE PACKAGE BODY MY\_PROC\_PACK IS

PROCEDURE ADAUGARE\_STOC\_PIESA( p\_id IN NUMBER, p\_cantitate IN NUMBER) IS

v\_stoc\_nou NUMBER;

BEGIN

SELECT stoc INTO v\_stoc\_nou

FROM dealer\_piese\_schimb

WHERE id\_piesa = p\_id;

v\_stoc\_nou:= v\_stoc\_nou + p\_cantitate;

UPDATE dealer\_piese\_schimb

SET stoc = v\_stoc\_nou

WHERE id\_piesa = p\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Piesa nu a fost gasita.');

END IF;

DBMS\_OUTPUT.PUT\_LINE('Stcocul piesei cu ID ' || p\_id || ' este: ' || v\_stoc\_nou);

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Eroare la actualizarea stocului.');

END;

PROCEDURE VALOARE\_INVENTAR\_MARCA(p\_marca IN VARCHAR2,p\_valoare OUT NUMBER) IS

BEGIN

SELECT SUM(pret) INTO p\_valoare

FROM dealer\_vehicule

WHERE UPPER(marca) = UPPER(p\_marca);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Marca specificata nu exista pe stoc.');

END;

PROCEDURE INSERARE\_CLIENT(p\_id IN NUMBER,p\_nume IN VARCHAR2, p\_prenume IN VARCHAR2, p\_telefon IN VARCHAR2, p\_email IN VARCHAR2) IS

BEGIN

INSERT INTO dealer\_clienti VALUES

(p\_id,p\_nume,p\_prenume,p\_telefon,p\_email);

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Clientul nu a putut fi inserat!');

END IF;

DBMS\_OUTPUT.PUT\_LINE('Client inserat cu succes: ' || p\_id || ' , ' || p\_nume || ' , ' || p\_prenume || ' , ' || p\_telefon || ' , ' || p\_email);

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20005, 'Eroare la inserarea clientului!');

END;

PROCEDURE STERGERE\_VEHICUL(p\_id IN NUMBER) IS

v\_cnt NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_cnt

FROM dealer\_piese\_schimb

WHERE id\_vehicul = p\_id;

IF v\_cnt > 0 THEN

RAISE\_APPLICATION\_ERROR(-20006, 'Vehiculul nu poate fi sters deoarece exista in dealer\_piese\_schimb.');

END IF;

SELECT COUNT(\*)

INTO v\_cnt

FROM dealer\_service

WHERE id\_vehicul = p\_id;

IF v\_cnt > 0 THEN

RAISE\_APPLICATION\_ERROR(-20007, 'Vehiculul nu poate fi sters deoarece exista in dealer\_service.');

END IF;

DELETE FROM dealer\_vehicule

WHERE id\_vehicul = p\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20008, 'Vehiculul nu a fost gasit.');

END IF;

DBMS\_OUTPUT.PUT\_LINE('Vehicul sters cu succes!');

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20009, 'Eroare la stergerea vehiculului.');

END;

END;

Apeluri:

DECLARE

v\_val NUMBER:=0;

BEGIN

my\_proc\_pack.adaugare\_stoc\_piesa(1, 10);

my\_proc\_pack.valoare\_inventar\_marca('Dacia',v\_val);

DBMS\_OUTPUT.PUT\_LINE('Valoare vehiculelor cu marca selectata: ' || v\_val);

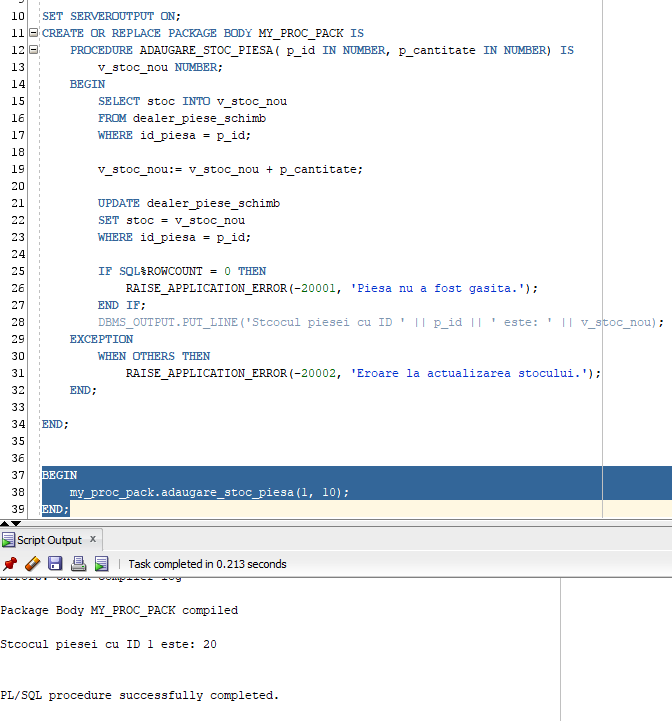
my\_proc\_pack.inserare\_client(6,'Ionut','Voicu','0799999999','voicudaniel22@stud.ase.ro');

my\_proc\_pack.stergere\_vehicul(6);

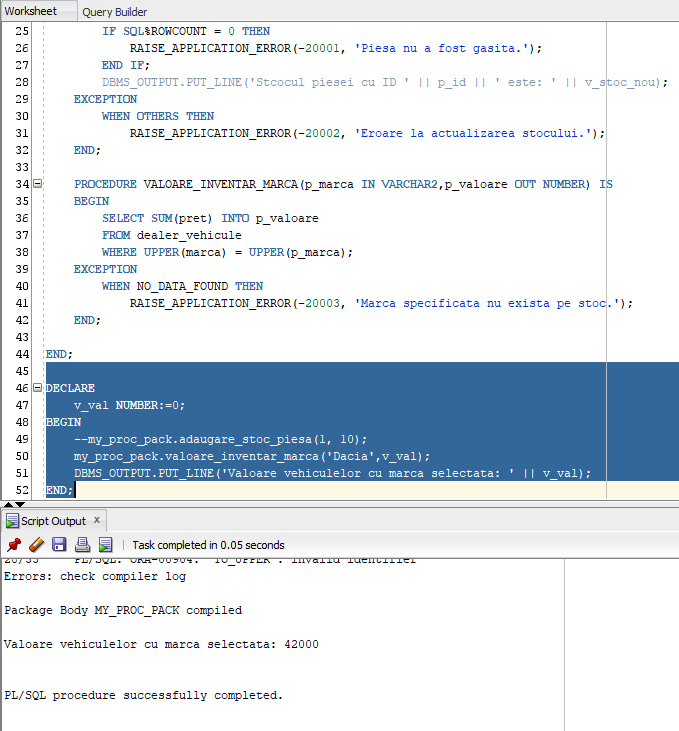
END;

Screenshot-uri:

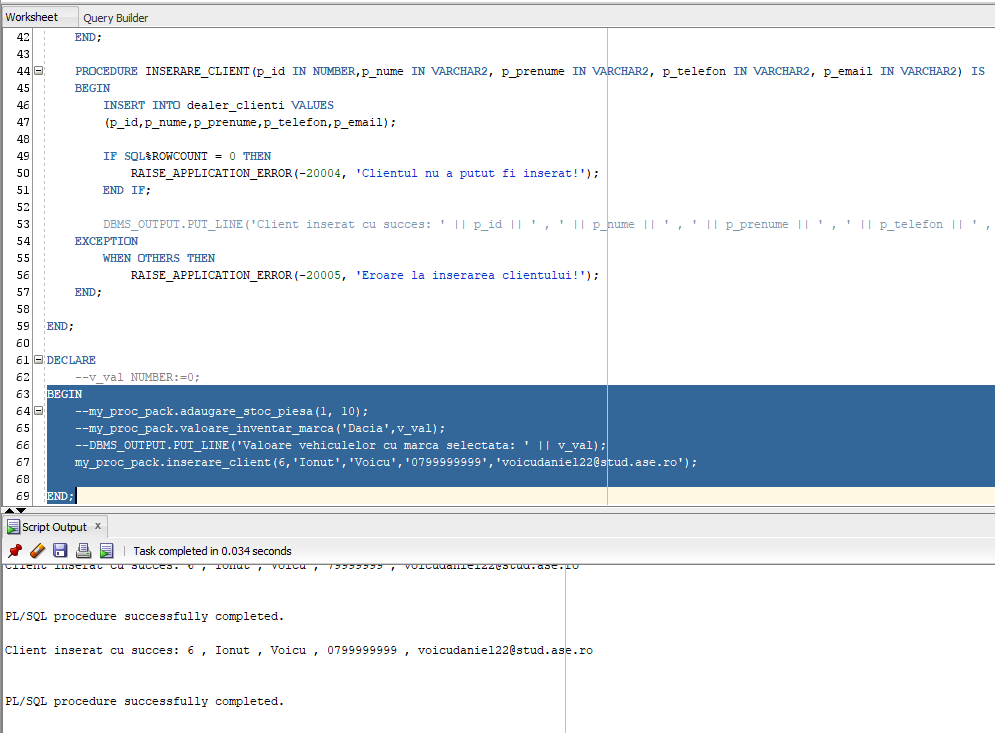
Adaugare\_stoc\_piesa:

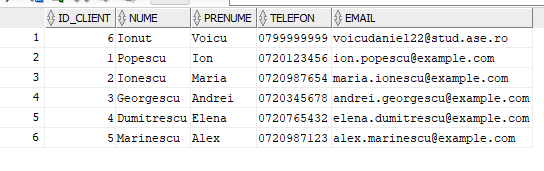


Valoare\_inventar\_marca:

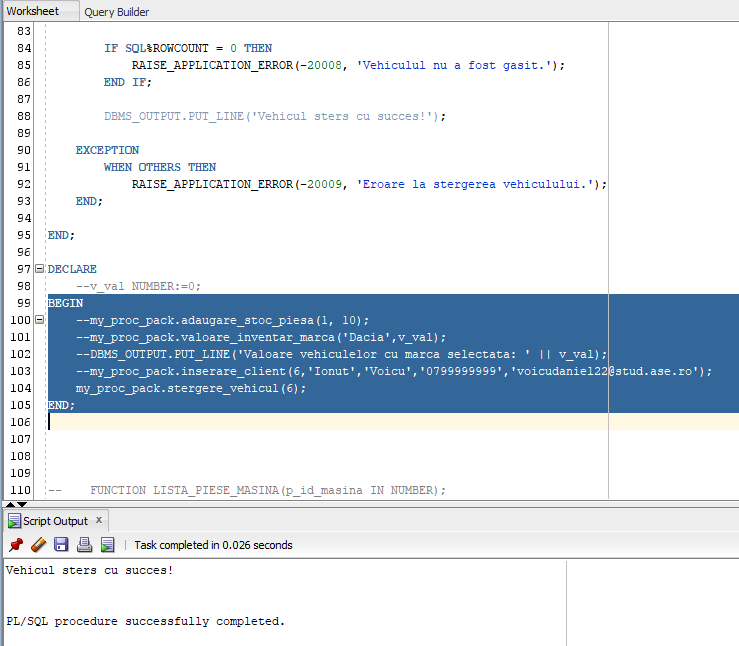


Inserare\_client:





Stergere\_vehicul:



Pachetul MY\_FUNC\_PACK – 3 functii

Antet:

CREATE OR REPLACE PACKAGE MY\_FUNC\_PACK IS

FUNCTION LISTA\_PIESE\_MASINA(p\_id\_masina IN NUMBER) RETURN VARCHAR2;

FUNCTION VERIFICA\_STOC\_PIESA(p\_id IN NUMBER) RETURN VARCHAR2;

FUNCTION PRET\_TOTAL\_PIESE\_VEHICUL(p\_id\_masina IN NUMBER) RETURN NUMBER;

END;

Body:

CREATE OR REPLACE PACKAGE BODY MY\_FUNC\_PACK IS

FUNCTION LISTA\_PIESE\_MASINA(p\_id\_masina IN NUMBER) RETURN VARCHAR2 IS

v\_lista\_piese VARCHAR2(4000);

CURSOR c\_piese IS

SELECT nume\_piesa

FROM dealer\_piese\_schimb

WHERE id\_vehicul = p\_id\_masina;

BEGIN

v\_lista\_piese := '';

FOR r\_piesa IN c\_piese LOOP

IF v\_lista\_piese IS NOT NULL THEN

v\_lista\_piese := v\_lista\_piese || ', ' || r\_piesa.nume\_piesa;

ELSE

v\_lista\_piese := r\_piesa.nume\_piesa;

END IF;

END LOOP;

IF v\_lista\_piese IS NULL THEN

v\_lista\_piese := 'Nu exista piese pentru acest vehicul.';

END IF;

RETURN v\_lista\_piese;

EXCEPTION

WHEN OTHERS THEN

RETURN 'Eroare la obtinerea listei de piese.';

END;

FUNCTION VERIFICA\_STOC\_PIESA(p\_id IN NUMBER) RETURN VARCHAR2 IS

v\_stoc NUMBER;

BEGIN

SELECT stoc

INTO v\_stoc

FROM dealer\_piese\_schimb

WHERE id\_piesa = p\_id;

IF v\_stoc > 0 THEN

RETURN 'Stoc piesa: ' || v\_stoc;

ELSE

RETURN 'Piesa nu este disponibila in stoc.';

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Piesa nu a fost gasita.';

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20010, 'Eroare la verificarea stocului piesei.');

END;

FUNCTION PRET\_TOTAL\_PIESE\_VEHICUL(p\_id\_masina IN NUMBER) RETURN NUMBER IS

v\_total NUMBER := 0;

BEGIN

SELECT SUM(pret)

INTO v\_total

FROM dealer\_piese\_schimb

WHERE id\_vehicul = p\_id\_masina;

RETURN NVL(v\_total, 0);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 0;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20011, 'Eroare la calcularea pretului total al pieselor.');

END;

END;

Apeluri:

DECLARE

v\_lista\_piese VARCHAR2(4000);

v\_id NUMBER := 1;

v\_ret VARCHAR2(500);

v\_total NUMBER := 0;

BEGIN

v\_lista\_piese := my\_func\_pack.lista\_piese\_masina(1);

DBMS\_OUTPUT.PUT\_LINE('Lista pieselor pentru masina cu ID ' || v\_id || ': ' || v\_lista\_piese);

v\_ret := my\_func\_pack.verifica\_stoc\_piesa(1);

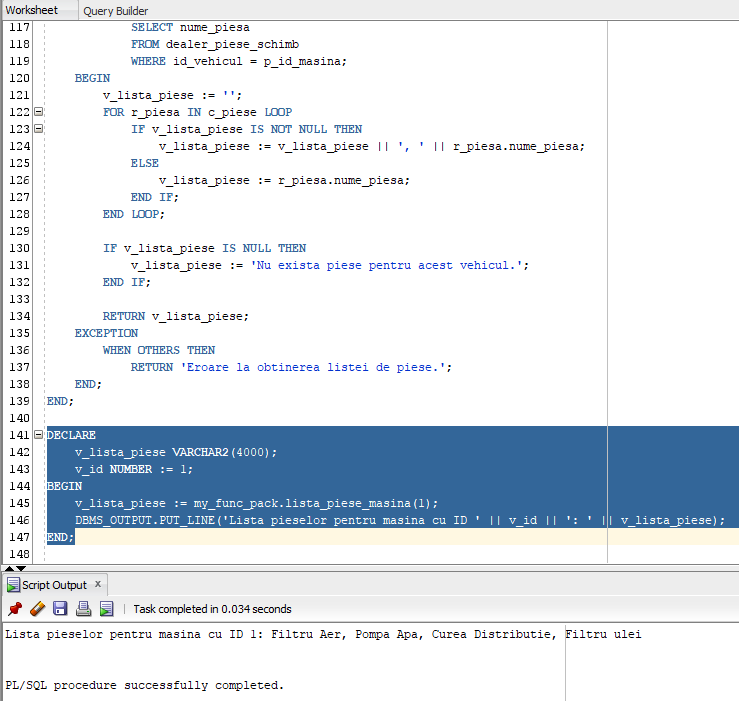
DBMS\_OUTPUT.PUT\_LINE(v\_ret);

v\_total := my\_func\_pack.pret\_total\_piese\_vehicul(1);

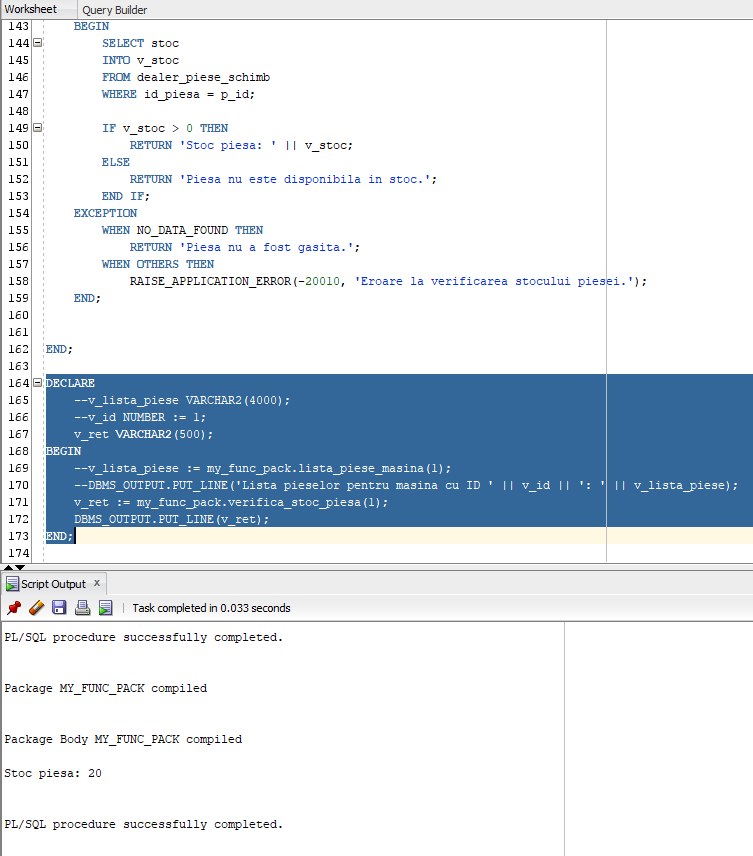
DBMS\_OUTPUT.PUT\_LINE(v\_total);

END;

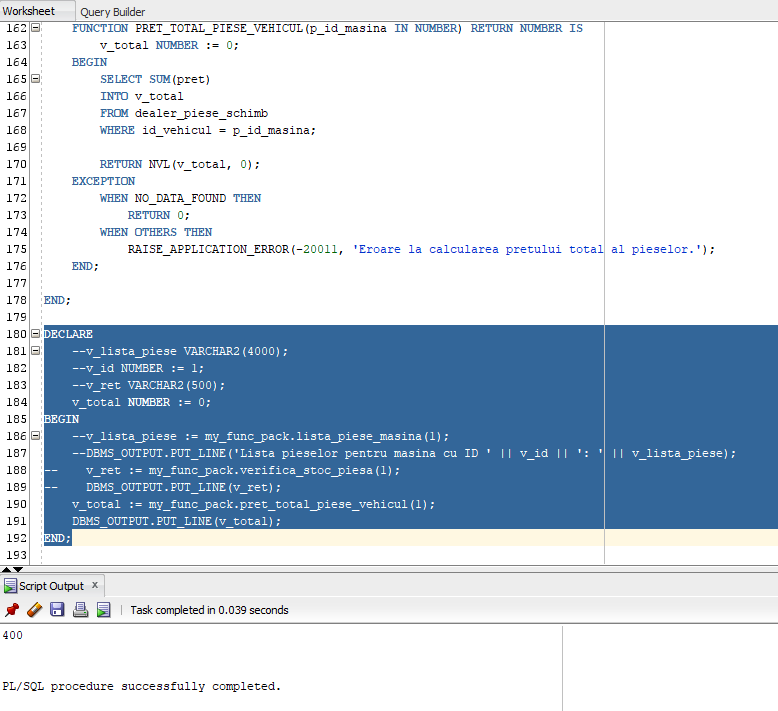
Screenshot-uri:

Lista\_piese\_masina:

Verifica\_stoc\_piesa:



Pret\_total\_piese\_vehicul:



# Triggers:

1. Trigger care actualizeaza automat id-ul unui vehicul inserat in BD:

CREATE OR REPLACE TRIGGER TRG\_PK\_VEHICUL

BEFORE INSERT ON DEALER\_VEHICULE

FOR EACH ROW

DECLARE

max\_id NUMBER;

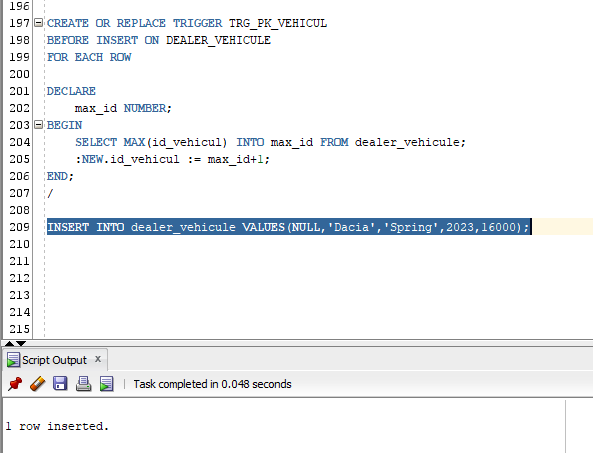
BEGIN

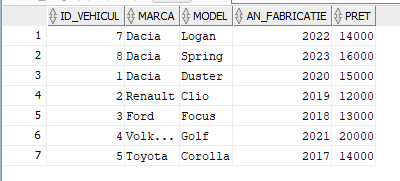
SELECT MAX(id\_vehicul) INTO max\_id FROM dealer\_vehicule;

:NEW.id\_vehicul := max\_id+1;

END;

INSERT INTO dealer\_vehicule VALUES(NULL,'Dacia','Spring',2023,16000);





1. Trigger care verifica existenta unui e-mail si nu permite inserarea de dubluri.

CREATE OR REPLACE TRIGGER trg\_duplicate\_email

BEFORE INSERT OR UPDATE ON dealer\_clienti

FOR EACH ROW

DECLARE

v\_cnt NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_cnt

FROM dealer\_clienti

WHERE email = :NEW.email AND id\_client != :NEW.id\_client;

IF v\_cnt > 0 THEN

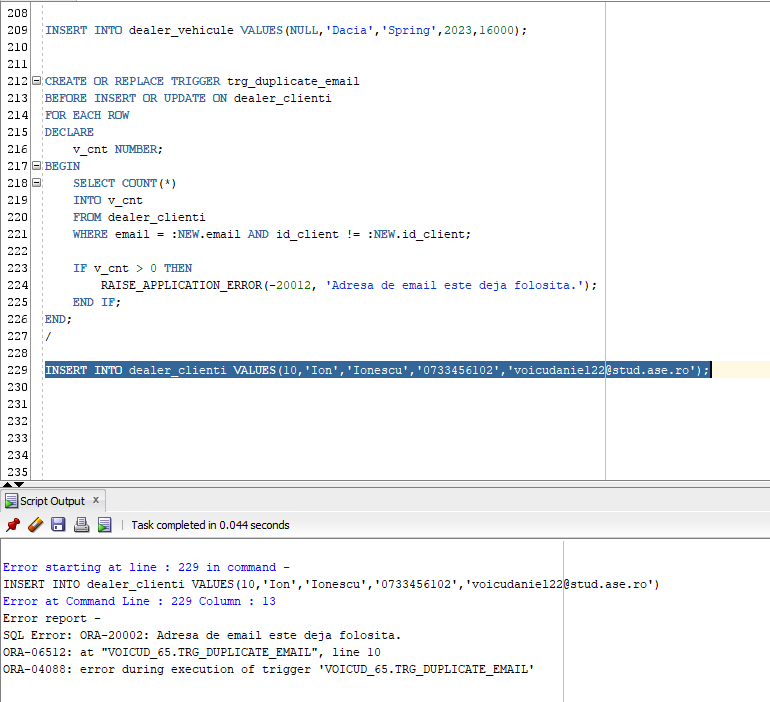
RAISE\_APPLICATION\_ERROR(-20012, 'Adresa de email este deja folosita.');

END IF;

END;

/

INSERT INTO dealer\_clienti VALUES(10,'Ion','Ionescu','0733456102','voicudaniel22@stud.ase.ro');



1. Trigger pentru adaugarea unei stergeri intr-o tabela noua (jurnal de stergeri)

* Am folosit acelasi trigger de la punctul 1. Pentru a incrementa automat ID-ul inregistrarii in tabela noua jurnal\_stergeri\_vehicule.

CREATE TABLE jurnal\_stergere\_vehicule (

id\_jurnal NUMBER PRIMARY KEY,

id\_vehicul NUMBER,

marca VARCHAR2(100),

model VARCHAR2(100),

an\_fabricatie NUMBER,

pret NUMBER,

data\_stergere DATE

);

CREATE OR REPLACE TRIGGER TRG\_PK\_JURNAL

BEFORE INSERT ON JURNAL\_STERGERE\_VEHICULE

FOR EACH ROW

DECLARE

max\_id NUMBER;

BEGIN

SELECT MAX(id\_jurnal) INTO max\_id FROM jurnal\_stergere\_vehicule;

:NEW.id\_jurnal := max\_id+1;

IF max\_id IS NULL THEN

:NEW.id\_jurnal := 1;

END IF;

END;

/

* Apoi am creat trigger-ul care creaza intrarea in tabela jurnal o data cu stergerea din tabela vehicule

CREATE OR REPLACE TRIGGER trg\_log\_stergere\_vehicul

AFTER DELETE ON dealer\_vehicule

FOR EACH ROW

BEGIN

INSERT INTO jurnal\_stergere\_vehicule (

id\_jurnal, id\_vehicul, marca, model, an\_fabricatie, pret, data\_stergere

) VALUES (

NULL, :OLD.id\_vehicul, :OLD.marca, :OLD.model, :OLD.an\_fabricatie, :OLD.pret, SYSDATE

);

END;

/

DELETE FROM dealer\_vehicule WHERE id\_vehicul = 7;

