**Academia de Studii Economice**

**Facultatea de Cibernetică, Statistică și Informatică Economică**

****

**PROIECT SGBD**

**MAGAZIN DE ECHIPAMENTE MEDICALE**

**Lungu Costin**

**București 2020**

Cuprins

[A) Descrierea problemei și prezentarea schemei conceptuale a tabelelor 3](#_Toc41339404)

[B) Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL 6](#_Toc41339405)

[C) STRUCTURI ALTERNATIVE 9](#_Toc41339406)

[D) Tratarea EXCEPȚIILOR predefinite (IMPLICITE) 11](#_Toc41339407)

[Tratarea excepțiilor definite de utilizator (EXPLICITE) 13](#_Toc41339408)

[E) CURSORI IMPLICIȚI: 15](#_Toc41339409)

[E) CURSORI EXPLICIȚI: 16](#_Toc41339410)

[F) Funcții, proceduri, includerea acestora în pachete 19](#_Toc41339411)

[Pachete: 22](#_Toc41339412)

[G) Declanșatori: 26](#_Toc41339413)

[Scripturi folosite 30](#_Toc41339414)

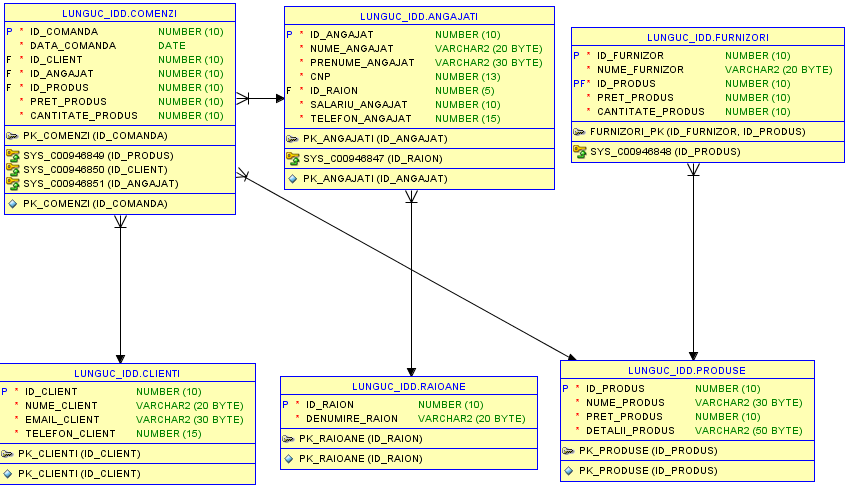
# Descrierea problemei și prezentarea schemei conceptuale a tabelelor

Am ales crearea unei baze de date pentru o firma de echipamente medicale, în vederea căruia am stocat date în 6 tabele, conectate între ele prin tipurile de legături dobândite în urma parcurgerii seminarelor.

Firma de echipamente medicale oferă clienților produse multiple ce sunt obținute în urma cumpărării de la mai mulți furnizori. În cadrul magazinului există angajați organizați pe mai multe raioane, în funcție de produsele specifice vândute, respectiv pe fizioterapie, cardiologie și estetică medicală.

Pentru ca magazinul de echipamente medicale să gestioneze fluxul de produse, a fost creată baza de date cu următoarele tabele și relații:

SCHIȚĂ TABELELOR BAZEI DE DATE

****

**TABELE EXISTENTE ȘI ÎMBUNĂTĂȚIȚE:**

Tabela ANGAJAȚI:

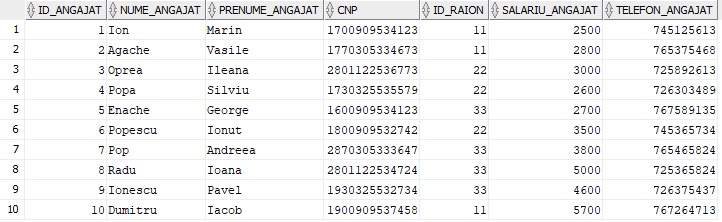


Tabela CLIENȚI:

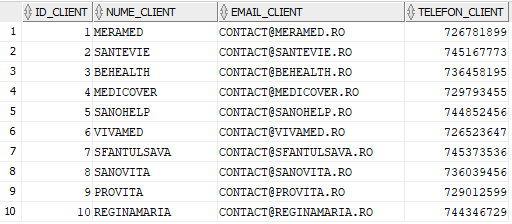


Tabela COMENZI:

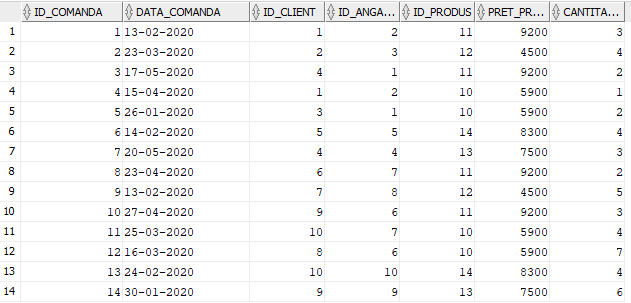


Tabela FURNIZORI:

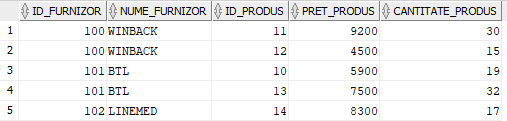


Tabela PRODUSE:

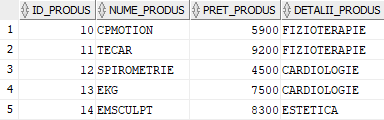
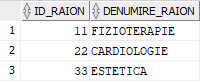


Tabela RAIOANE:



# Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL

Crearea tabelei ESTETICA in care au fost introduși angajații din raionul estetica:

SET SERVEROUTPUT ON

begin

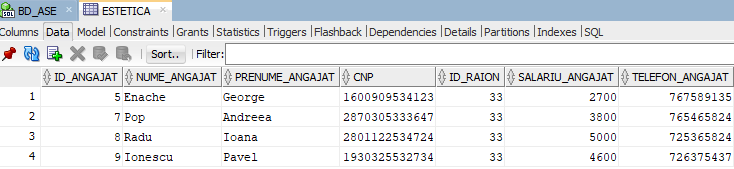
EXECUTE IMMEDIATE

'CREATE TABLE ESTETICA AS SELECT \*

FROM ANGAJATI

WHERE ID\_RAION=(33) ';

end;



Inserarea în tabela Produse un nou produs:

BEGIN

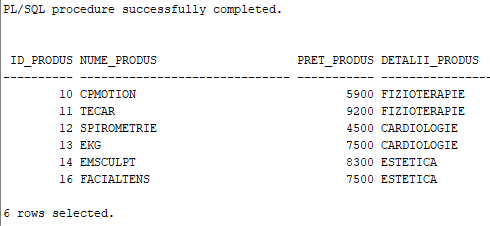
INSERT INTO PRODUSE (id\_produs, nume\_produs, pret\_produs, detalii\_produs)

VALUES (16, 'FACIALTENS', 7500, 'ESTETICA');

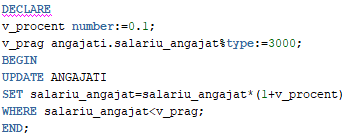
END;

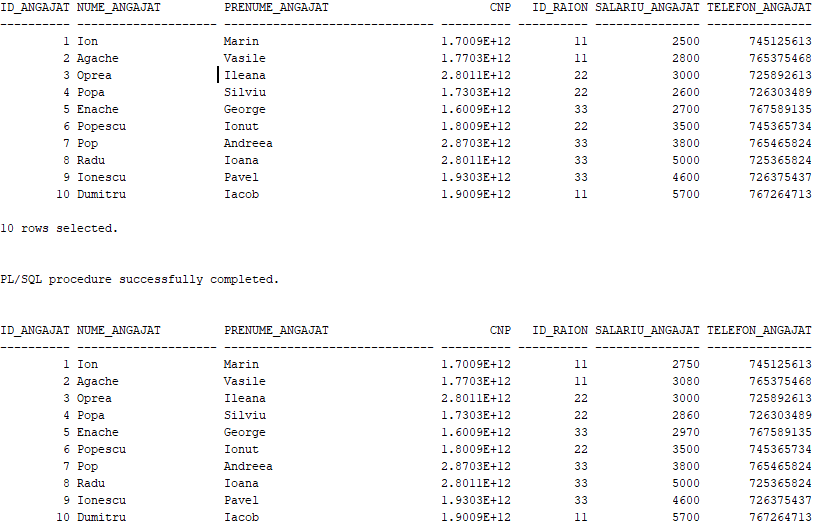
/

select \* from PRODUSE;

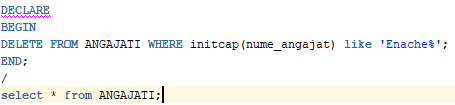


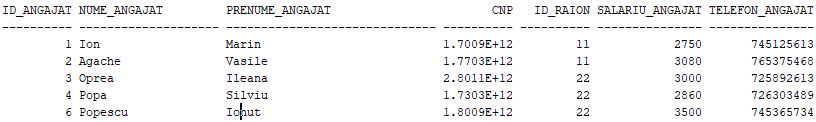
Folosind comanda UPDATE se mărește cu 10% salariul angajaților din tabela ANGAJATI care au în prezent salariul mai mic decât 3000 de lei:





Folosind comanda DELETE, angajatul cu numele Enache este sters din tabela ANGAJATI:





Afișarea id-ului și numelui clientului care a cumpărat 5 produse per comandă:

SET SERVEROUTPUT ON

DECLARE

v\_id number(5);

v\_nume varchar2(20);

BEGIN

Select c.ID\_CLIENT,c.NUME\_CLIENT

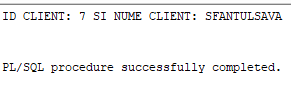
INTO v\_id, v\_nume

from CLIENTI c, COMENZI CO

where c.ID\_CLIENT=CO.ID\_CLIENT and CO.CANTITATE\_PRODUS=5;

DBMS\_OUTPUT.PUT\_LINE('ID CLIENT: '||v\_id ||' SI NUME CLIENT: '||v\_nume);

END;



Am folosit variabila de substitutie v\_nume, comanda PRINT pentru introducerea de la tastatură a numelui unui angajat și afișarea salariului acestuia:

SET SERVEROUTPUT ON

DECLARE

v\_nume angajati.nume\_angajat%type;

BEGIN

SELECT nume\_angajat, salariu\_angajat into v\_nume, :SALARIU

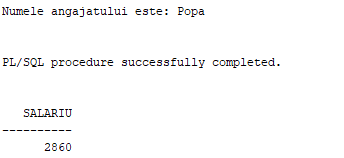
FROM ANGAJATI WHERE NUME\_ANGAJAT='&Nume';

DBMS\_OUTPUT.PUT\_LINE ('Numele angajatului este: '||v\_nume);

END;

/

print SALARIU



# STRUCTURI ALTERNATIVE

Folosind structura „**IF.. THEN.. END IF**” au fost afișate prețurile produselor cu id-urile 10 și 11, adăugându-se la prețul de initial procentul de 15%, dacă produsul avea prețul cuprins între 5000 și 7000 de lei, respectiv procentul de 10%, dacă produsul avea prețul mai mare de 7000:

SET SERVEROUTPUT ON

DECLARE

v\_initial produse.pret\_produs%type;

BEGIN

SELECT pret\_produs into v\_initial from produse where id\_produs=&Produs;

dbms\_output.put\_line ('Pretul initial este: '|| v\_initial);

IF v\_initial < 5000 THEN

v\_initial:=1.2\* v\_initial;

ELSIF v\_initial between 5000 and 7000 THEN

v\_initial:=1.15 \* v\_initial;

ELSE

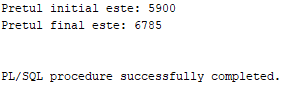
v\_initial:=1.1\* v\_initial;

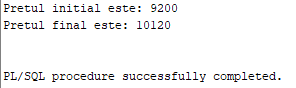
END IF;

dbms\_output.put\_line('Pretul final este: '|| v\_initial);

end;

/





Folosind structura „**CASE… WHEN… THEN…**” a fost calculată reducerea comenzilor în funcție de cantitatea de produse cumpărate, respective pentru comenzi de 1-2 produse reducerea a fost de 10%, între 3-5 produse cumpărate o reducere de 15%, iar peste 5 produse o reducere de 20% din valoarea totală a comenzii:

SET SERVEROUTPUT ON

DEFINE p;

ACCEPT p PROMPT 'Introduceti id comanda:';

DECLARE

cant comenzi.cantitate\_produs%TYPE;

pret comenzi.pret\_produs%type;

BEGIN

SELECT cantitate\_produs, pret\_produs into cant, pret from comenzi

where id\_comanda=&p;

dbms\_output.put\_line ('Pretul comenzii este: '|| pret\*cant);

dbms\_output.put\_line ('Cantitatea produselor cumpărate: '|| cant);

CASE

WHEN cant < 3 THEN

pret:= 0.1 \* pret \* cant;

WHEN cant between 3 and 5 THEN

pret:= 0.15 \* pret \* cant;

ELSE

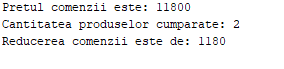
pret:= 0.2 \* pret \* cant;

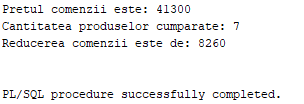
END CASE;

dbms\_output.put\_line('Reducerea comenzii este de: '|| pret);

end;

/





**C)** Structuri repetitive

Folosind structura **„WHILE…..LOOP….END LOOP”**a fost calculată media salariului angajaților, afișându-se angajații cu salariul mai mic decât media

SET SERVEROUTPUT ON

DECLARE

ID\_ANGAJAT angajati.id\_angajat%type;

NUME angajati.nume\_angajat%type;

SALARIU angajati.salariu\_angajat%TYPE;

MEDIA angajati.salariu\_angajat%TYPE;

I NUMBER(3):=1;

BEGIN

SELECT AVG(salariu\_angajat)INTO MEDIA FROM angajati;

DBMS\_OUTPUT.PUT\_LINE('MEDIA SALARIILOR ANGAJATILOR ESTE: '||MEDIA);

WHILE I<=10 LOOP

SELECT nume\_angajat, salariu\_angajat INTO NUME, SALARIU FROM angajati

WHERE id\_angajat=I;

DBMS\_OUTPUT.PUT\_LINE('ANGAJATUL CU ID '||I|| ' ARE NUMELE '||NUME|| ' SI SALARIUL DE '||SALARIU);

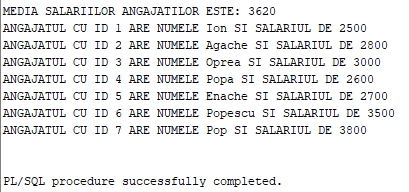
I:=I+1;

exit when SALARIU>MEDIA;

END LOOP;

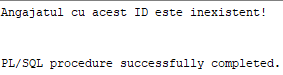
END;

/



# Tratarea EXCEPȚIILOR predefinite (IMPLICITE)

Folosind EXCEPȚIILE predefinite ale Serverului Oracle s-a încercat efectuarea afișării angajatului cu id-ul 13; acesta fiind inexistând am primit afișarea:



SET SERVEROUTPUT ON

DECLARE

v\_nume VARCHAR2(20);

BEGIN

SELECT nume\_angajat INTO v\_nume

FROM angajati

WHERE id\_angajat=13;

dbms\_output.put\_line(v\_nume);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Angajatul cu acest ID este inexistent!');

END;

/

Folosind EXCEPȚIILE predefinite ale Serverului Oracle s-a efectuat afișarea angajatului cu prenumele Marin și a numărului de telefon, presupunând că este onomastica acestuia, iar compania dorește să îl felicite, astfel încât la apelul funcției a apărut o eroare, întrucât în cadrul firmei există mai mulți salariați cu același nume, fiind afișată EXCEPȚIA conform căreia „Exista mai multi angajati cu prenumele de Marin”:

SET SERVEROUTPUT ON

DECLARE

tel angajati.telefon\_angajat%type;

BEGIN

select telefon\_angajat into tel from angajati where prenume\_angajat='Marin';

DBMS\_OUTPUT.PUT\_LINE('Marin are numarul de telefon: '||tel);

EXCEPTION

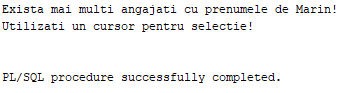
WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Exista mai multi angajati cu prenumele de Marin!

Utilizati un cursor pentru selectie!');

END;

/



Folosind apelul „open c” se deschide un CURSOR, ulterior încercând parcurgerea sa folosind „for”. Instrucțiunea „for” încearcă să deschidă din nou cursorul rezultând excepția

CURSOR\_ALREADY\_OPEN:

DECLARE

cursor c is select nume\_angajat,prenume\_angajat,salariu\_angajat from angajati order by salariu\_angajat desc;

BEGIN

open c;

for r in c loop

exit when c%rowcount>5;

dbms\_output.put\_line(r.nume\_angajat||' '||r.prenume\_angajat||' '||r.salariu\_angajat);

end loop;

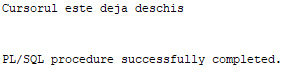
EXCEPTION

WHEN CURSOR\_ALREADY\_OPEN then

dbms\_output.put\_line('Cursorul este deja deschis');

end;

/



**Tratarea EXCEPȚIILOR non-predefinite Oracle Server**

Folosind EXCEPȚIILE non-predefinite Oracle Server a fost tratată situația în care introducerea în tabela „RAIOANE” a unui nou raion cu numele „ORTOPEDIE”, fără a-i fi precizat id-ul acestuia:

SET SERVEROUTPUT ON

DECLARE

INSERT\_EXCEPT EXCEPTION;

PRAGMA EXCEPTION\_INIT(INSERT\_EXCEPT, -01400);

BEGIN

insert into raioane (id\_raion, denumire\_raion) values (NULL, 'ORTOPEDIE');

EXCEPTION

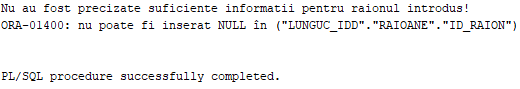
WHEN insert\_except THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost precizate suficiente informatii pentru raionul introdus!');

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/



## **Tratarea excepțiilor definite de utilizator (EXPLICITE)**

Folosind EXCEPȚIILE definite de utilizator au fost tratate situațiile în care denumirea produsului cu id-ul 3 este inexistend, deoarece produsele sunt create începând cu id-ul 10, respectiv s-a afișat o altă excepție prin care utilizator constată că s-a produs o eroare, în cazul de față s-a confundat „nume\_produs” cu „id\_produs”:

DECLARE

invalid\_prod EXCEPTION;

BEGIN

UPDATE produse

SET nume\_produs='Ortoap'

WHERE nume\_produs=3;

IF SQL%NOTFOUND THEN

RAISE invalid\_prod;

END IF;

EXCEPTION

WHEN invalid\_prod THEN

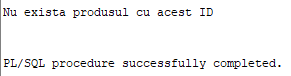
DBMS\_OUTPUT.PUT\_LINE('Nu exista produsul cu acest ID');

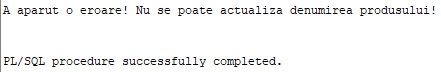
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare! Nu se poate actualiza denumirea produsului!');

END;

/





Am creat tabela Erori, definită de utilizator, în care au fost introduse excepțiile cu codul și mesajul de eroare din exemplul precedent:

CREATE TABLE ERORI(

utilizator varchar2(32),

data\_exc date,

cod\_exc number(7),

mesaj\_exc varchar2(128));

După aceea rulăm blocul PL/SQL:

SET SERVEROUTPUT ON

DECLARE

cod NUMBER(7);

mesaj VARCHAR2(255);

invalid\_prod EXCEPTION;

PRAGMA EXCEPTION\_INIT(invalid\_prod,-20999);

BEGIN

UPDATE produse

SET nume\_produs='Laptop ABC'

WHERE id\_produs=3;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR (-20999,'Cod produs invalid!');

END IF;

EXCEPTION

WHEN invalid\_prod THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produsul cu acest ID');

cod:=SQLCODE;

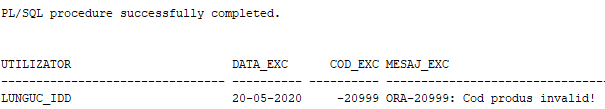
mesaj:=SQLERRM;

INSERT INTO ERORI VALUES(USER, SYSDATE, cod, mesaj);

END;

/

SELECT \* FROM ERORI;



Folosind exceptia pre-definită a fost invocată în mod explicit următoarea excepție:

DECLARE

invalid\_prod EXCEPTION;

BEGIN

UPDATE produse

SET nume\_produs='Laptop ABC'

WHERE id\_produs=3;

IF SQL%NOTFOUND THEN

RAISE NO\_DATA\_FOUND;

END IF;

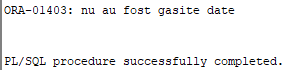
EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/



# CURSORI IMPLICIȚI:

Se șterg produsele din categoria „Fizioterapie”, care au fost comandate. Se afișează numărul de rânduri șterse:

SET SERVEROUTPUT ON

BEGIN

DELETE FROM produse p

WHERE DETALII\_PRODUS='FIZIOTERAPIE' and exists (select 1 from COMENZI C where p.id\_produs=C.id\_produs);

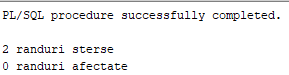
DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri sterse');

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri afectate');

END;

/



Se șterge din tabela PRODUSE, regiunea a cărei ID este introdus de utilizator prin intermediul variabilei de substituție g\_rid. Mesajul este afișat folosind variabila de mediu nr\_sters.

ACCEPT g\_rid PROMPT 'Introduceti id-ul produsului:'

VARIABLE nr\_sters varchar2(100)

DECLARE

BEGIN

DELETE FROM produse WHERE id\_produs=&g\_rid;

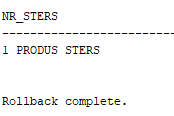
:nr\_sters:=TO\_CHAR(SQL%ROWCOUNT)||' PRODUS STERS';

END;

/

PRINT nr\_sters

ROLLBACK;



# CURSORI EXPLICIȚI:

Să se afişeze lista cu numele şi salariul angajaţilor din raionul 22, folosind un cursor explicit și trei variabile scalare:

set serveroutput on

DECLARE

cursor ang\_cursor is select id\_angajat, nume\_angajat, salariu\_angajat from angajati where id\_raion=22;

ang\_id angajati.id\_angajat%type;

ang\_nume angajati.nume\_angajat%type;

ang\_sal angajati.salariu\_angajat%type;

BEGIN

dbms\_output.put\_line('Lista cu salariariile angajatilor din raionul 22:');

open ang\_cursor;

loop

fetch ang\_cursor into ang\_id, ang\_nume, ang\_sal;

exit when ang\_cursor%notfound;

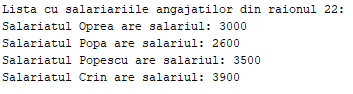
dbms\_output.put\_line('Salariatul '||ang\_nume||' are salariul: '||ang\_sal);

end loop;

close ang\_cursor;

end;

/



Să se încarce în tabela MESAJE, nou creată, primii 5 angajaţi (id şi nume):

CREATE TABLE mesaje

(cod varchar2(7),

nume varchar2(20)

);

DECLARE

v\_id angajati.id\_angajat%type;

v\_nume angajati.nume\_angajat%type;

CURSOR c1 IS SELECT id\_angajat, nume\_angajat FROM angajati;

BEGIN

OPEN c1;

FOR i IN 1..5 LOOP

FETCH c1 INTO v\_id, v\_nume;

INSERT INTO mesaje VALUES(v\_id, v\_nume);

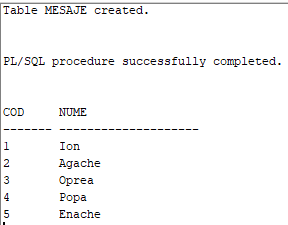
END LOOP;

CLOSE c1;

END;

/

SELECT \* FROM mesaje;



Se afişează printr-un ciclu FOR numele şi salariile angajaţilor din raionul 22:

set serveroutput on

declare

cursor ang\_cursor is select id\_angajat, nume\_angajat, salariu\_angajat from angajati where id\_raion=22;

begin

dbms\_output.put\_line('Lista cu salariariile angajatilor din raionul 22:');

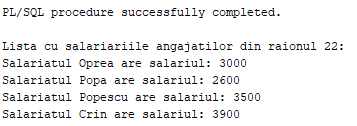
for ang\_rec in ang\_cursor loop

dbms\_output.put\_line('Salariatul '||ang\_rec.nume\_angajat||' are salariul: '||ang\_rec.salariu\_angajat);

end loop;

end;

/



**Cursor cu parametri:**

Să se afişeze produsele al căror valoarea totală comandată este mai mare decât valoarea totală a comenzilor pentru produsul 13:

SET SERVEROUTPUT ON

DECLARE

CURSOR c\_prod (p\_val NUMBER) IS

SELECT p.id\_produs, p.nume\_produs, sum(c.cantitate\_produs\*c.pret\_produs) total

FROM produse p, comenzi c

WHERE p.id\_produs =c.id\_produs

GROUP BY p.id\_produs, p.nume\_produs

HAVING sum(c.cantitate\_produs\*c.pret\_produs)>p\_val

ORDER BY total desc;

v\_val NUMBER(5);

rec\_prod c\_prod%rowtype;

BEGIN

select sum(pret\_produs\*cantitate\_produs) into v\_val from comenzi where id\_produs=13;

DBMS\_OUTPUT.PUT\_LINE('Produsele al caror cantitate vândută este mai mare decat '|| v\_val||' sunt:');

IF NOT c\_prod%ISOPEN THEN

OPEN c\_prod (v\_val);

END IF;

LOOP

FETCH c\_prod into rec\_prod;

EXIT WHEN c\_prod%notfound;

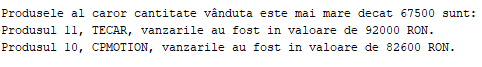
DBMS\_OUTPUT.PUT\_LINE('Produsul '||rec\_prod.id\_produs||', '||rec\_prod.nume\_produs||', vanzarile au fost in valoare de ' ||rec\_prod.total||' RON.');

END LOOP;

CLOSE c\_prod;

END;

/



# F) Funcții, proceduri, includerea acestora în pachete

**Proceduri:**

Procedura modifica\_salariul primeşte doi parametrii: p\_id\_angajat şi procent şi majorează cu procentul specificat salariul angajatului cu id\_angajat=p\_id\_angajat:

CREATE OR REPLACE

PROCEDURE modifica\_salariu\_angajat

(p\_id\_angajat IN angajati.id\_angajat%type, procent IN number)

IS

v\_salariul angajati.salariu\_angajat%type;

BEGIN

Select salariu\_angajat into v\_salariul from angajati where id\_angajat=p\_id\_angajat;

dbms\_output.put\_line('Angajatul are salariul de '||v\_salariul);

Update angajati

Set salariu\_angajat=salariu\_angajat\*(1+procent/100)

Where id\_angajat=p\_id\_angajat;

Select salariu\_angajat into v\_salariul from angajati where id\_angajat=p\_id\_angajat;

Dbms\_output.put\_line('Angajatul are acum salariul de '||v\_salariul);

END;

/

show errors;



Sa se realizeze apelul procedurii modifica\_salariu\_angajat:

SET SERVEROUTPUT ON

EXECUTE modifica\_salariu\_angajat(1, 14)

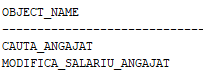


Sa se vizualizeze procedurile din dictionarul metadatelor:

select \*

from user\_objects

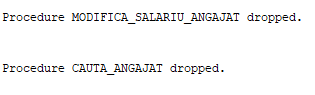
where object\_type='PROCEDURE';



Sa se stearga procedura modifica\_salariu\_angajat, cauta\_angajat:

DROP PROCEDURE modifica\_salariu\_angajat;

DROP PROCEDURE cauta\_angajat;



Procedura primeşte ca parametru de tip IN id\_ul unui angajat şi returnează prin parametrii de tip OUT numele şi salariul acestuia:

CREATE OR REPLACE PROCEDURE cauta\_angajat

(p\_id\_angajat IN angajati.id\_angajat%type,

p\_nume OUT angajati.nume\_angajat%type,

p\_salariul OUT angajati.salariu\_angajat%type)

IS

BEGIN

Select nume\_angajat, salariu\_angajat into p\_nume, p\_salariul from angajati where id\_angajat=p\_id\_angajat;

DBMS\_OUTPUT.PUT\_LINE(' Angajatul '||p\_nume||' are salariul de: '||p\_salariul);

END;

/



Sa se creeze o functie PL/SQL pentru calculul taxei tva:

CREATE OR REPLACE FUNCTION taxa (value IN NUMBER, proc IN NUMBER)

RETURN NUMBER IS

BEGIN

RETURN (value\*proc);

END taxa;

/

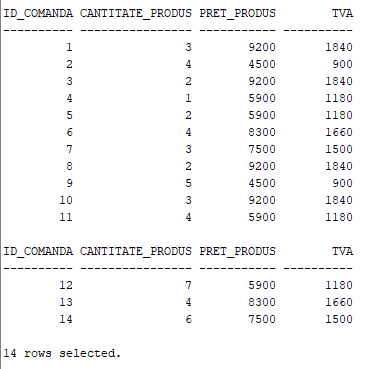
show errors



Apelul funcţiei:

SELECT id\_comanda, cantitate\_produs, pret\_produs, taxa (pret\_produs, 20) as tva

FROM comenzi;



CREATE OR REPLACE FUNCTION verifica\_client

(p\_nume\_client IN clienti.nume\_client%type)

RETURN Boolean

IS

v\_nume clienti.nume\_client%type;

BEGIN

SELECT nume\_client into v\_nume from clienti where nume\_client=p\_nume\_client;

IF v\_nume = p\_nume\_client then

return true;

ELSE

return false;

end if;

EXCEPTION

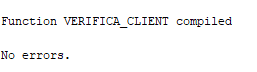
WHEN no\_data\_found THEN

return NULL;

end;

/

show errors



# Pachete:

Crearea pachetului cu specificarea subprogramelor:

create or replace PACKAGE actualizare\_produse IS

procedure adauga\_produs

(p\_id produse.id\_produs%type,

p\_nume produse.nume\_produs%type,

p\_pret produse.pret\_produs%type,

p\_det produse.detalii\_produs%type);

function cauta\_produs

(p\_nume produse.nume\_produs%type)

return number;

procedure modifica\_produs

(p\_id produse.id\_produs%type,

p\_nume produse.nume\_produs%type,

p\_pret produse.pret\_produs%type);

function sterge\_produs

(p\_id produse.id\_produs%type)

return boolean;

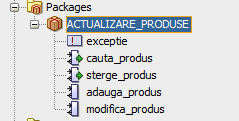
exceptie exception;

END;

/

show errors





Ștergerea corpului unui pachet:





Crearea unui **corp de pachet**, în care au fost introduse procedurile adauga\_produs, modifica\_produs și funcțiile cauta\_produs, respectiv sterge\_produs:

create or replace PACKAGE BODY actualizare\_produse IS

procedure adauga\_produs

(p\_id produse.id\_produs%type,

p\_nume produse.nume\_produs%type,

p\_pret produse.pret\_produs%type,

p\_det produse.detalii\_produs%type)

is

begin

insert into produse(id\_produs,nume\_produs,pret\_produs,detalii\_produs) values (p\_id,p\_nume,p\_pret,p\_det);

exception

when others then

raise exceptie;

end adauga\_produs;

function cauta\_produs

(p\_nume produse.nume\_produs%type)

return number

is

v\_id\_produs produse.id\_produs%type;

begin

select id\_produs into v\_id\_produs from produse

where nume\_produs like p\_nume||'%';

return v\_id\_produs;

exception

when no\_data\_found then

return null;

when too\_many\_rows then

return null;

end cauta\_produs;

procedure modifica\_produs

(p\_id produse.id\_produs%type,

p\_nume produse.nume\_produs%type,

p\_pret produse.pret\_produs%type)

is

begin

update produse

set nume\_produs=p\_nume, pret\_produs=p\_pret

where id\_produs=p\_id;

if sql%notfound then

raise exceptie;

end if;

end modifica\_produs;

function sterge\_produs

(p\_id produse.id\_produs%type)

return boolean

is

begin

delete from produse

where id\_produs=p\_id;

if sql%found then

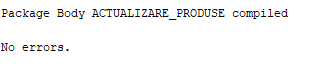
raise exceptie;

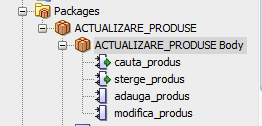
end if;

end sterge\_produs;

END;

/



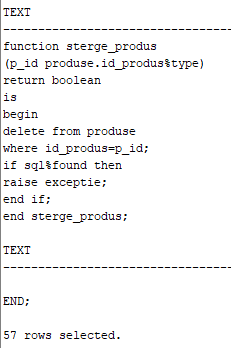


Vizualizarea ultimei părți a corpului de pachet:

Select text

From user\_source

Where name='ACTUALIZARE\_PRODUSE' and type='PACKAGE BODY';



# G) Declanșatori:

Se creează un trigger care se declanşează înaintea fiecărei operaţii de inserare în tabela PRODUSE.

CREATE OR REPLACE TRIGGER produse\_trig

BEFORE INSERT ON produse

BEGIN

dbms\_output.put\_line('triggerul s-a executat');

END;

/



Ștergerea declanșatorului restrict\_salaraiul:

DROP TRIGGER restrict\_salariul



Am creat triggerul produse\_trig\_log, care se declanşează la operaţiile de INSERT, DELETE sau UPDATE pe tabela Produse. În tabela TEMP\_LOG se introduce tipul operaţiei, utilizatorul care a executat-o și data curentă:

CREATE TABLE temp\_log

(tip CHAR(1),

utilizator VARCHAR2(50),

data DATE DEFAULT SYSDATE);

CREATE OR REPLACE TRIGGER produse\_trig\_log

BEFORE INSERT or UPDATE or DELETE on produse

DECLARE

v\_tip temp\_log.tip%TYPE;

BEGIN

case

when INSERTING then v\_tip :='I';

when UPDATING then v\_tip:='U';

ELSE v\_tip :='D';

END case;

INSERT INTO temp\_log(tip, utilizator, data) VALUES (v\_tip, user, sysdate);

END;

/



Inserarea în tabela, ștergerea din tabelă și afișarea tabelei declanșatorilor:

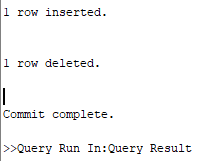
insert into produse (id\_produs, nume\_produs,pret\_produs,detalii\_produs)

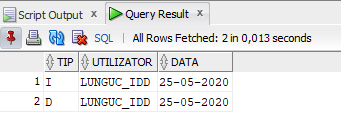
values (15, 'TECAR-PRO',9999,'FIZIOTERAPIE');

delete from produse where id\_produs=15;

commit;

Select \* from temp\_log;





Am creat un declanșator care asigură unicitatea codului produsului folosind valorile generate de o secvenţă

CREATE SEQUENCE produse\_secv

START WITH 1

INCREMENT BY 1

MAXVALUE 100

NOCYCLE;

CREATE OR REPLACE TRIGGER generare\_codprodus

BEFORE INSERT ON produse

FOR EACH ROW

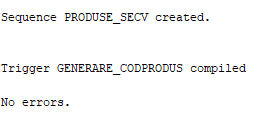
BEGIN

SELECT produse\_secv.nextval INTO :new.id\_produs FROM dual;

END;

/

show errors;



Crearea unei tabele virtuale:

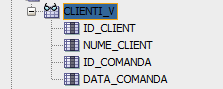
create or replace view clienti\_v as

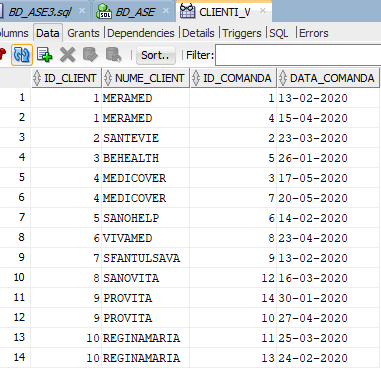
select cl.id\_client, cl.nume\_client, co.id\_comanda, co.data\_comanda

from clienti cl, comenzi co

where cl.id\_client=co.id\_client;







Creare unui trigger, folosind clauzele inserting, updating, deleting, la comanda cărora se declanșează:

create or replace trigger trigger\_COMENZI

instead of insert or update or delete on clienti\_v

for each row

begin

if inserting then

insert into clienti (id\_client, nume\_client) values (:new.id\_client, :new.nume\_client);

insert into comenzi (id\_comanda, data\_comanda, id\_client) values (:new.id\_comanda, :new.data\_comanda, :new.id\_client);

elsif deleting then

delete from comenzi where id\_comanda=:old.id\_comanda;

elsif updating ('nume\_client') then

update clienti

set nume\_client=:new.nume\_client

where id\_client=:old.id\_client;

elsif updating ('data\_comanda') then

update comenzi

set data\_comanda=:new.data\_comanda

where id\_comanda=:old.id\_comanda;

end if;

end;

/

show errors;

# Scripturi folosite

SET SERVEROUTPUT ON

begin

EXECUTE IMMEDIATE

'CREATE TABLE ESTETICA AS SELECT \*

FROM ANGAJATI

WHERE ID\_RAION=(33) ';

end;

BEGIN

INSERT INTO PRODUSE (id\_produs, nume\_produs, pret\_produs, detalii\_produs)

VALUES (16, 'FACIALTENS', 7500, 'ESTETICA');

END;

/

select \* from PRODUSE;

DECLARE

v\_procent number:=0.1;

v\_prag angajati.salariu\_angajat%type:=3000;

BEGIN

UPDATE ANGAJATI

SET salariu\_angajat=salariu\_angajat\*(1+v\_procent)

WHERE salariu\_angajat<v\_prag;

END;

/

select \* from ANGAJATI;

DECLARE

BEGIN

DELETE FROM ANGAJATI WHERE initcap(nume\_angajat) like 'Enache%';

END;

/

select \* from ANGAJATI;

SET SERVEROUTPUT ON

DECLARE

v\_id number(5);

v\_nume varchar2(20);

BEGIN

Select c.ID\_CLIENT,c.NUME\_CLIENT

INTO v\_id, v\_nume

from CLIENTI c, COMENZI CO

where c.ID\_CLIENT=CO.ID\_CLIENT and CO.CANTITATE\_PRODUS=5;

DBMS\_OUTPUT.PUT\_LINE('ID CLIENT: '||v\_id ||' SI NUME CLIENT: '||v\_nume);

END;

SET SERVEROUTPUT ON

DECLARE

v\_nume angajati.nume\_angajat%type;

BEGIN

SELECT nume\_angajat, salariu\_angajat into v\_nume, :SALARIU

FROM ANGAJATI WHERE NUME\_ANGAJAT='&Nume';

DBMS\_OUTPUT.PUT\_LINE ('Numele angajatului este: '||v\_nume);

END;

/

print SALARIU

SET SERVEROUTPUT ON

DECLARE

v\_initial produse.pret\_produs%type;

BEGIN

SELECT pret\_produs into v\_initial from produse where id\_produs=&Produs;

dbms\_output.put\_line ('Pretul initial este: '|| v\_initial);

IF v\_initial < 5000 THEN

v\_initial:=1.2\* v\_initial;

ELSIF v\_initial between 5000 and 7000 THEN

v\_initial:=1.15 \* v\_initial;

ELSE

v\_initial:=1.1\* v\_initial;

END IF;

dbms\_output.put\_line('Pretul final este: '|| v\_initial);

end;

/

SET SERVEROUTPUT ON

DEFINE p;

ACCEPT p PROMPT 'Introduceti id comanda:';

DECLARE

cant comenzi.cantitate\_produs%TYPE;

pret comenzi.pret\_produs%type;

BEGIN

SELECT cantitate\_produs, pret\_produs into cant, pret from comenzi

where id\_comanda=&p;

dbms\_output.put\_line ('Pretul comenzii este: '|| pret\*cant);

dbms\_output.put\_line ('Cantitatea produselor cumpărate: '|| cant);

CASE

WHEN cant < 3 THEN

pret:= 0.1 \* pret \* cant;

WHEN cant between 3 and 5 THEN

pret:= 0.15 \* pret \* cant;

ELSE

pret:= 0.2 \* pret \* cant;

END CASE;

dbms\_output.put\_line('Reducerea comenzii este de: '|| pret);

end;

/