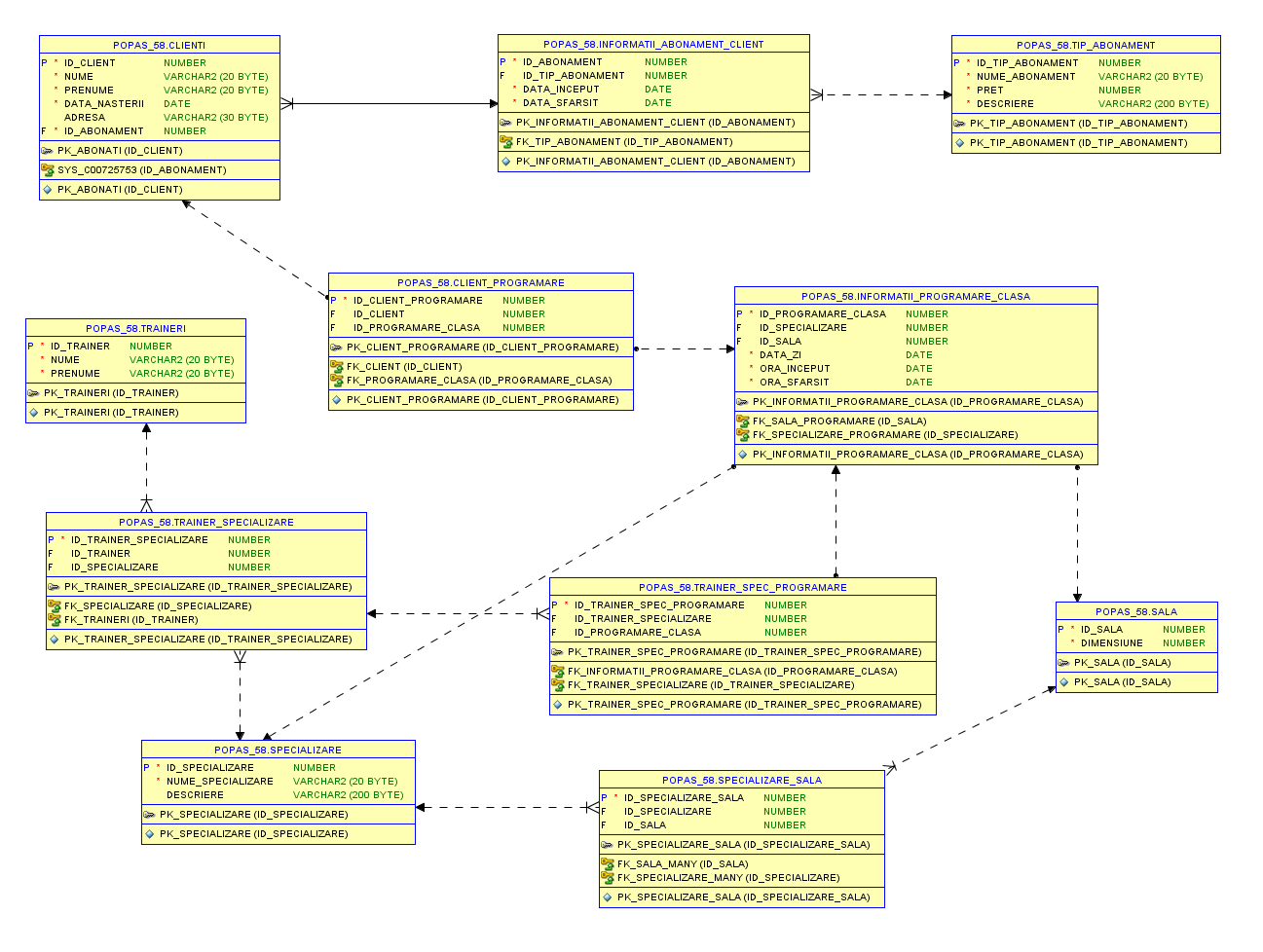
# Baze de date – Sala de fitness

Aceasta baza de date este dedicata unei sali de fitness. Aceasta stocheaza informatii despre fiecare client, tinand cont de ofertele de abonamente ale salii de fitness si implementeaza un sistem de gestiune al orarului pentru clasele de fitness. Pentru o programare a unei clase se tine de cont de faptul ca pot fii mai multi traineri pentru clasa respectiva, de specializarea de referinta pentru clasa, daca sala este dotata pentru specificul specializarii si de a nu se suprapune clasele intre ele.



create table clienti

(

id\_client number constraint pk\_abonati primary key,

nume varchar2(20) not null,

prenume varchar2(20) not null,

data\_nasterii date not null,

adresa varchar2(30),

id\_abonament number not null

);

create table tip\_abonament

(

id\_tip\_abonament number constraint pk\_tip\_abonament primary key,

nume\_abonament varchar2(20) not null,

pret number not null,

descriere varchar2(200) not null

);

create table informatii\_abonament\_client

(

id\_abonament number constraint pk\_informatii\_abonament\_client primary key,

id\_tip\_abonament number constraint fk\_tip\_abonament references tip\_abonament(id\_tip\_abonament),

data\_inceput date not null,

data\_sfarsit date not null

);

ALTER TABLE clienti

ADD FOREIGN KEY(id\_abonament) REFERENCES informatii\_abonament\_client(id\_abonament);

create table traineri

(

id\_trainer number constraint pk\_traineri primary key,

nume varchar2(20) not null,

prenume varchar2(20) not null,

specializari varchar2(20) not null

);

create table specializare

(

id\_specializare number constraint pk\_specializare primary key,

nume\_specializare varchar2(20) not null,

descriere varchar2(200)

);

create table trainer\_specializare

(

id\_trainer\_specializare number constraint pk\_trainer\_specializare primary key,

id\_trainer number constraint fk\_traineri references traineri(id\_trainer),

id\_specializare number constraint fk\_specializare references specializare(id\_specializare)

);

create table sala

(

id\_sala number constraint pk\_sala primary key,

dimensiune number not null

);

create table informatii\_programare\_clasa

(

id\_programare\_clasa number constraint pk\_informatii\_programare\_clasa primary key,

id\_specializare number constraint fk\_specializare\_programare references specializare(id\_specializare),

id\_sala number constraint fk\_sala\_programare references sala(id\_sala),

data\_zi date not null,

ora\_inceput date not null,

ora\_sfarsit date not null

);

create table client\_programare

(

id\_client\_programare number constraint pk\_client\_programare primary key,

id\_client number constraint fk\_client references clienti(id\_client),

id\_programare\_clasa number constraint fk\_programare\_clasa references informatii\_programare\_clasa(id\_programare\_clasa)

);

create table trainer\_spec\_programare

(

id\_trainer\_spec\_programare number constraint pk\_trainer\_spec\_programare primary key,

id\_trainer\_specializare number constraint fk\_trainer\_specializare references trainer\_specializare(id\_trainer\_specializare),

id\_programare\_clasa number constraint fk\_informatii\_programare\_clasa references informatii\_programare\_clasa(id\_programare\_clasa)

);

-- identifier too long error

create table specializare\_sala

(

id\_specializare\_sala number constraint pk\_specializare\_sala primary key,

id\_specializare number constraint fk\_specializare\_many references specializare(id\_specializare),

id\_sala number constraint fk\_sala\_many references sala(id\_sala)

);

alter table traineri

drop column specializari;

-- inserturi

insert into tip\_abonament(id\_tip\_abonament,nume\_abonament,pret, descriere)

values(1,'Day time',120,'acces zilnic pana la ora 16');

insert into tip\_abonament(id\_tip\_abonament,nume\_abonament,pret, descriere)

values(2,'Full time',200,'acces la orice ora');

insert into tip\_abonament(id\_tip\_abonament,nume\_abonament,pret, descriere)

values(3,'Full time',220,'acces la orice ora');

insert into informatii\_abonament\_client(id\_abonament, id\_tip\_abonament, data\_inceput, data\_sfarsit)

values(1,1,to\_date('13/01/2020','dd/mm/yyyy'),to\_date('12/02/2020','dd/mm/yyyy'));

insert into informatii\_abonament\_client(id\_abonament, id\_tip\_abonament, data\_inceput, data\_sfarsit)

values(2,2,to\_date('16/03/2020','dd/mm/yyyy'),to\_date('15/04/2020','dd/mm/yyyy'));

insert into informatii\_abonament\_client(id\_abonament, id\_tip\_abonament, data\_inceput, data\_sfarsit)

values(3,3,to\_date('16/03/2020','dd/mm/yyyy'),to\_date('15/04/2020','dd/mm/yyyy'));

insert into clienti(id\_client,nume,prenume,data\_nasterii,adresa,id\_abonament)

values(1,'Cristian','Dinca',to\_date('03/11/2001', 'DD/MM/YYYY'),'str mihail moxa 2',2);

insert into clienti(id\_client,nume,prenume,data\_nasterii,adresa,id\_abonament)

values(2,'Alexandru','Ioan',to\_date('09/03/1999', 'DD/MM/YYYY'),'str mihail moxa 11',1);

insert into clienti(id\_client,nume,prenume,data\_nasterii,adresa,id\_abonament)

values(3,'Costel','Ioan',to\_date('19/05/1999', 'DD/MM/YYYY'),'str alexandru cel bun 10',1);

insert into clienti(id\_client,nume,prenume,data\_nasterii,adresa,id\_abonament)

values(4,'Vasilian','Ion',to\_date('20/11/1999', 'DD/MM/YYYY'),'str alexandru cel bun 7',1);

insert into clienti(id\_client,nume,prenume,data\_nasterii,adresa,id\_abonament)

values(5,'Cristian','Cornel',to\_date('03/12/2005', 'DD/MM/YYYY'),'str mihail moxa 2',3);

insert into specializare(id\_specializare, nume\_specializare, descriere)

values(1,'antrenor personal','ajutor pentru atingerea obiectivelor sportive personale');

insert into specializare(id\_specializare, nume\_specializare, descriere)

values(2,'yoga','Practica yoga are ca scop flexibilitatea corpul, linistirea mintii si recapatarea starii de bine in intregul corp');

insert into specializare(id\_specializare, nume\_specializare, descriere)

values(3,'cardio','Este o specializare ce presupune exercitii ce stimuleaza frecventa cardiaca, fiind suficient de antrenanta');

insert into traineri(id\_trainer, nume, prenume)

values(1,'Chiriac','Vlad');

insert into traineri(id\_trainer, nume, prenume)

values(2,'Marinescu','Oana');

insert into traineri(id\_trainer, nume, prenume)

values(3,'Cristian','Ionita');

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(1,1,1);

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(2,1,2);

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(3,1,3);

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(4,2,1);

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(5,3,1);

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(6,3,3);

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(7,2,3);

insert into sala(id\_sala, dimensiune)

values(1,40);

insert into sala(id\_sala, dimensiune)

values(2,70);

insert into specializare\_sala(id\_specializare\_sala,id\_sala,id\_specializare)

values(1,1,2);

insert into specializare\_sala(id\_specializare\_sala,id\_sala,id\_specializare)

values(2,1,3);

insert into specializare\_sala(id\_specializare\_sala,id\_sala,id\_specializare)

values(3,2,3);

insert into informatii\_programare\_clasa(id\_programare\_clasa, id\_specializare, id\_sala, data\_zi, ora\_inceput, ora\_sfarsit)

values(1,3,1,to\_date('18/03/2020', 'DD/MM/YYYY'),to\_date('15:30', 'HH24:MI'),to\_date('17:30', 'HH24:MI'));

insert into informatii\_programare\_clasa(id\_programare\_clasa, id\_specializare, id\_sala, data\_zi, ora\_inceput, ora\_sfarsit)

values(2,3,2,to\_date('18/03/2020', 'DD/MM/YYYY'),to\_date('16:20', 'HH24:MI'),to\_date('17:45', 'HH24:MI'));

insert into trainer\_spec\_programare(id\_trainer\_spec\_programare,id\_trainer\_specializare,id\_programare\_clasa)

values(1,3,1);

insert into trainer\_spec\_programare(id\_trainer\_spec\_programare,id\_trainer\_specializare,id\_programare\_clasa)

values(2,7,1);

insert into trainer\_spec\_programare(id\_trainer\_spec\_programare,id\_trainer\_specializare,id\_programare\_clasa)

values(3,6,2);

insert into client\_programare(id\_client\_programare,id\_client,id\_programare\_clasa)

values(1,1,1);

insert into client\_programare(id\_client\_programare,id\_client,id\_programare\_clasa)

values(2,2,1);

insert into client\_programare(id\_client\_programare,id\_client,id\_programare\_clasa)

values(3,4,1);

insert into client\_programare(id\_client\_programare,id\_client,id\_programare\_clasa)

values(4,3,2);

--1. Reduceti abonamentul cu 10% clientilor nascuti in luna “decembrie”

update tip\_abonament

set pret=pret-0.1\*pret

where (select t.id\_tip\_abonament from clienti c, informatii\_abonament\_client i, tip\_abonament t

where t.id\_tip\_abonament=i.id\_tip\_abonament and c.id\_abonament=i.id\_abonament and extract(month from c.data\_nasterii)=12)=tip\_abonament.id\_tip\_abonament;

-- 2. Reduceti cu 10 locuri dimensiunea salilor care au specializarea “yoga”

update sala

set dimensiune =dimensiune -10

where dimensiune>10 and

(select s.id\_sala from specializare\_sala s, sala, specializare sp where s.id\_sala=sala.id\_sala and s.id\_specializare=sp.id\_specializare and sp.nume\_specializare='yoga')=sala.id\_sala;

--3. Mutati pe 3 decembrie 2020 programarile de clasa unde participa mai mult de 2 clienti

update informatii\_programare\_clasa

set data\_zi=to\_date('03/12/2020', 'dd/mm/yyyy')

where (select i.id\_programare\_clasa from client\_programare c, informatii\_programare\_clasa i

where c.id\_programare\_clasa=i.id\_programare\_clasa

group by i.id\_programare\_clasa

having count(c.id\_client)>=2)=informatii\_programare\_clasa.id\_programare\_clasa;

--4. Modificati prenumele trainer-ului cu id-ul 2 in “Cristian”

update traineri

set prenume='Cristian'

where id\_trainer=2;

--5. Modificati adresa clientilor care au numele “Cristian” si care sunt nascuti dupa anul 2002 in "SUA"

update clienti

set adresa='SUA'

where extract(year from data\_nasterii)>2002 and nume='Cristian';

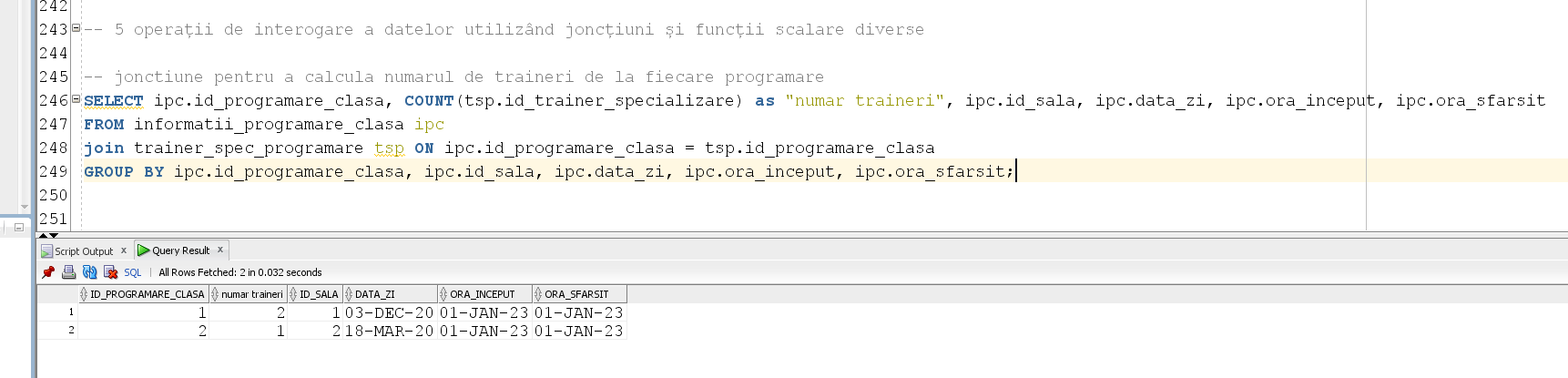
-- 5 operații de interogare a datelor utilizând joncțiuni și funcții scalare diverse

**-- jonctiune pentru a calcula numarul de traineri de la fiecare programare**

SELECT ipc.id\_programare\_clasa, COUNT(tsp.id\_trainer\_specializare) as "numar traineri", ipc.id\_sala, ipc.data\_zi, ipc.ora\_inceput, ipc.ora\_sfarsit

FROM informatii\_programare\_clasa ipc

join trainer\_spec\_programare tsp ON ipc.id\_programare\_clasa = tsp.id\_programare\_clasa

GROUP BY ipc.id\_programare\_clasa, ipc.id\_sala, ipc.data\_zi, ipc.ora\_inceput, ipc.ora\_sfarsit; 

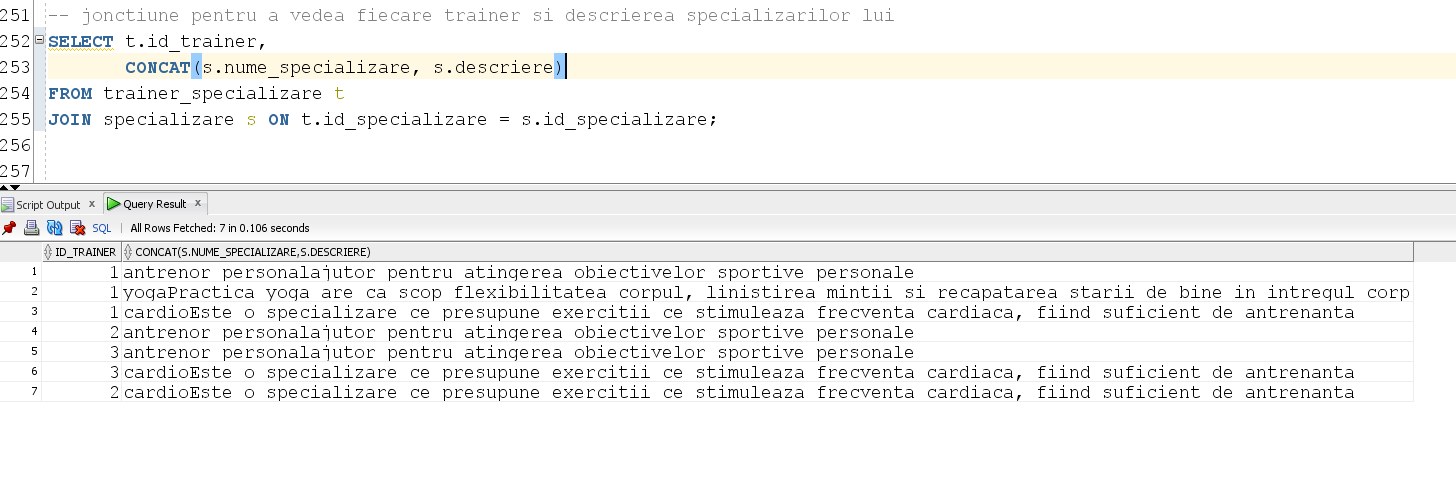
**-- jonctiune pentru a vedea fiecare trainer si descrierea specializarilor lui**

SELECT t.id\_trainer,

CONCAT(s.nume\_specializare, s.descriere)

FROM trainer\_specializare t

JOIN specializare s ON t.id\_specializare = s.id\_specializare;

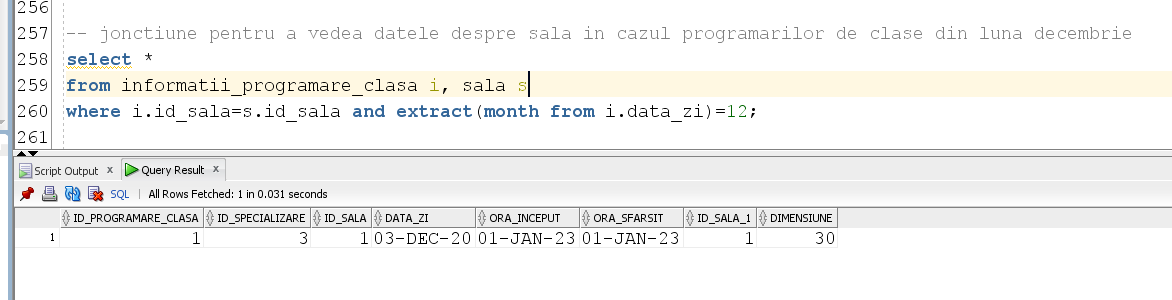


**-- jonctiune pentru a vedea datele despre sala in cazul programarilor de clase din luna decembrie**

select \*

from informatii\_programare\_clasa i, sala s

where i.id\_sala=s.id\_sala and extract(month from i.data\_zi)=12;



Semestrul 2

## **TEMA - Proiect SGBD, v.1**

Utilizând schema BD de la proiect (cel din semestrul 1, care va fi continuat în acest semestru), construiți cel puțin 5 blocuri PL/SQL care să conțină:

* structuri de control
* cursori impliciți
* cursori expliciți

Notă: se punctează complexitatea blocurilor create; acestea ar trebui să includă **simultan**cât mai multe dintre mecanismele învățate: structuri de control, cursori...

-- 1.Se afiseaza informatii despre clientul cu ID-ul dat

dbms\_output.enable(buffer\_size IN INTEGER DEFAULT 20000);

exec dbms\_output.enable(1000000);

SET SERVEROUTPUT ON

DECLARE

client\_id clienti.id\_client%TYPE :='&id' ;

client\_nume clienti.nume%TYPE;

client\_prenume clienti.prenume%TYPE;

abonament\_nume tip\_abonament.nume\_abonament%TYPE;

abonament\_descriere tip\_abonament.descriere%TYPE;

BEGIN

SELECT nume, prenume INTO client\_nume, client\_prenume FROM clienti WHERE id\_client = client\_id;

IF SQL%FOUND=TRUE THEN

DBMS\_OUTPUT.PUT\_LINE('Nume client: ' || client\_nume || ' ' || client\_prenume);

SELECT nume\_abonament, descriere INTO abonament\_nume, abonament\_descriere

FROM tip\_abonament JOIN informatii\_abonament\_client ON

tip\_abonament.id\_tip\_abonament = informatii\_abonament\_client.id\_tip\_abonament

WHERE id\_abonament = (SELECT id\_abonament FROM clienti WHERE id\_client = client\_id);

DBMS\_OUTPUT.PUT\_LINE('Tip abonament: ' || abonament\_nume);

DBMS\_OUTPUT.PUT\_LINE('Descriere abonament: ' || abonament\_descriere);

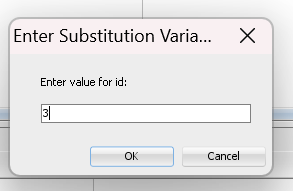
ELSE

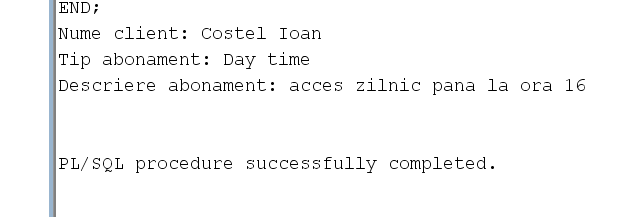
DBMS\_OUTPUT.PUT\_LINE('Nu exista clientul cu id ' || client\_id);

END IF;

END;

/





-- 2. Sa se afiseze numele si descrierea tuturor tipurilor de abonamente

DECLARE

v\_nume\_abonament tip\_abonament.nume\_abonament%TYPE;

v\_descriere tip\_abonament.descriere%TYPE;

v\_pret tip\_abonament.pret%TYPE;

BEGIN

FOR i IN (SELECT nume\_abonament, descriere, pret FROM tip\_abonament)

LOOP

v\_nume\_abonament := i.nume\_abonament;

v\_descriere := i.descriere;

v\_pret := i.pret;

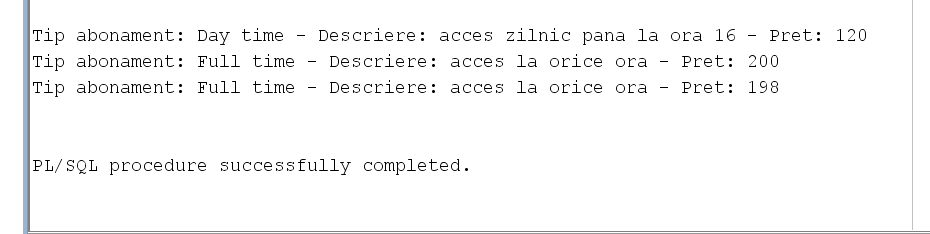
DBMS\_OUTPUT.PUT\_LINE('Tip abonament: ' || v\_nume\_abonament || ' - Descriere: ' ||

v\_descriere || ' - Pret: ' || v\_pret);

END LOOP;

END;

/



-- 3. Afiseaza numele si specializarea tuturor trainerilor

DECLARE

CURSOR c\_traineri IS

SELECT t.nume, s.nume\_specializare

FROM traineri t

JOIN trainer\_specializare ts ON t.id\_trainer = ts.id\_trainer

JOIN specializare s ON ts.id\_specializare = s.id\_specializare;

BEGIN

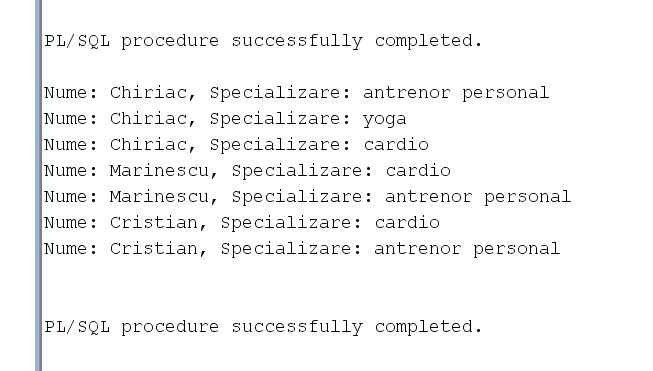
FOR trainer IN c\_traineri LOOP

DBMS\_OUTPUT.PUT\_LINE('Nume: ' || trainer.nume || ', Specializare: ' || trainer.nume\_specializare);

END LOOP;

END;

/



-- 4. Afiseaza clientii majori

DECLARE

CURSOR majori IS

SELECT id\_client, nume, prenume, data\_nasterii, adresa

FROM clienti

WHERE months\_between(sysdate, data\_nasterii)/12 >= 18;

BEGIN

FOR i IN majori

LOOP

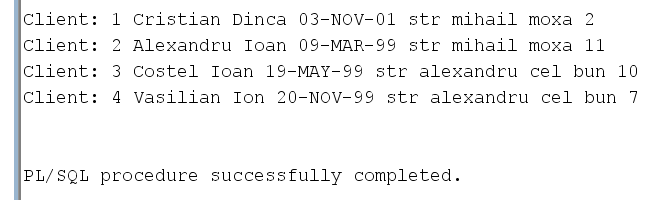
DBMS\_OUTPUT.PUT\_LINE('Client: ' || i.id\_client || ' ' || i.nume|| ' '|| i.prenume

|| ' ' || i.data\_nasterii || ' ' || i.adresa);

END LOOP;

END;

/



-- 5. Modifica adresa clientului cu ID=1 si afiseaza numarul de modificari efectuate

DECLARE

v\_num\_rows\_updated INTEGER;

BEGIN

UPDATE clienti

SET adresa = '&adresa'

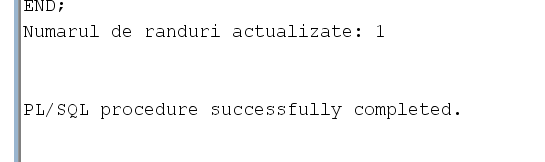
WHERE id\_client = 1;

v\_num\_rows\_updated := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Numarul de randuri actualizate: ' || v\_num\_rows\_updated);

END;

/



## **TEMA - Proiect SGBD, v.2**

Utilizând schema BD de la proiect (cel din semestrul 1, care va fi continuat în acest semestru), construiți cel puțin 4 blocuri PL/SQL care să conțină:

* excepții implicite (de sistem)
* excepții explicite (definite de utilizator)

Nota: se punctează complexitatea blocurilor create; acestea ar trebui sa includă simultan cât mai multe dintre mecanismele învățate: structuri de control, cursori, excepții...

----- 1.Bloc PL/SQL pentru actualizarea numelui unui client in tabela clienti

DECLARE

v\_id\_client NUMBER := &id;

verificare\_id NUMBER;

v\_nume VARCHAR2(20) := '&nume\_update';

BEGIN

SELECT id\_client INTO verificare\_id FROM CLIENTI WHERE v\_id\_client= clienti.id\_client;

UPDATE clienti SET nume = v\_nume WHERE id\_client = v\_id\_client;

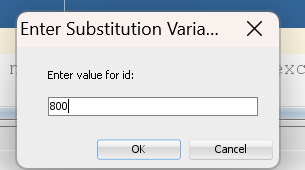
EXCEPTION

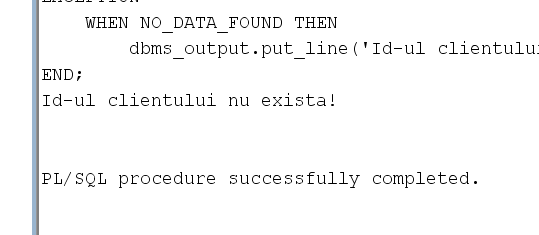
WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Id-ul clientului nu exista!');

END;

/





--2. bloc pl/sql care sa afiseze primii 3 cei mai batrani clienti, si daca nu exista sa se scrie o exceptie personalizata

DECLARE

CURSOR c\_clienti IS

SELECT id\_client, nume, prenume, data\_nasterii

FROM clienti

ORDER BY data\_nasterii DESC

FETCH FIRST 3 ROWS ONLY;

v\_id\_client clienti.id\_client%TYPE;

v\_nume clienti.nume%TYPE;

v\_prenume clienti.prenume%TYPE;

v\_data\_nasterii clienti.data\_nasterii%TYPE;

v\_count INTEGER:=0;

clienti\_insuficienti EXCEPTION;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM clienti; -- numara numarul de clienti

IF v\_count < 3 THEN -- verifica daca exista mai putin de 3 clienti

RAISE clienti\_insuficienti;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Primii 3 cei mai batrani clienti:');

FOR r\_clienti IN c\_clienti LOOP

v\_id\_client := r\_clienti.id\_client;

v\_nume := r\_clienti.nume;

v\_prenume := r\_clienti.prenume;

v\_data\_nasterii := r\_clienti.data\_nasterii;

DBMS\_OUTPUT.PUT\_LINE(v\_nume || ' ' || v\_prenume || ' - Data nasterii: ' || v\_data\_nasterii);

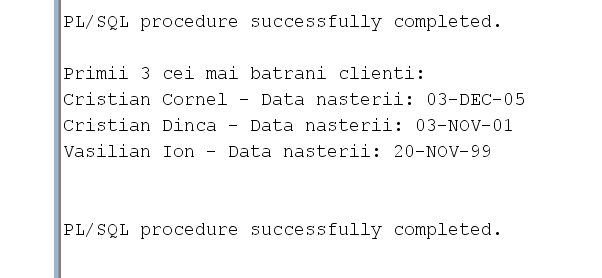
END LOOP;

EXCEPTION

WHEN clienti\_insuficienti THEN DBMS\_OUTPUT.PUT\_LINE('Clienti insuficienti!');

END;

/



-- 3. doar specializarile existente în tabela "specializare" pot fi atribuite unui trainer:

DECLARE

trainer\_specializare\_exception EXCEPTION;

cheie\_duplicata EXCEPTION;

---00001 CODUL ERORII PT ACEASTA EXCEPTIE

v\_verif NUMBER;

v\_id\_trainer\_specializare trainer\_specializare.id\_trainer\_specializare%TYPE;

v\_id\_trainer trainer\_specializare.id\_trainer%TYPE;

v\_id\_specializare trainer\_specializare.id\_specializare%TYPE;

PRAGMA EXCEPTION\_INIT(cheie\_duplicata,-00001);

BEGIN

FOR i IN 1..&num\_values LOOP

v\_id\_trainer\_specializare := &id\_trainer\_specializare;

v\_id\_trainer := &id\_trainer;

v\_id\_specializare := &id\_specializare;

BEGIN

SELECT id\_specializare INTO v\_verif FROM specializare s WHERE s.id\_specializare = v\_id\_specializare;

INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

VALUES (v\_id\_trainer\_specializare, v\_id\_trainer, v\_id\_specializare);

DBMS\_OUTPUT.PUT\_LINE(“Row inserted successfully.”);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(“Specializarea nu exista in tabela "specializare”);

WHEN cheie\_duplicata THEN

DBMS\_OUTPUT.PUT\_LINE(“Cheie duplicata”);

END;

END LOOP;

COMMIT;

END;

/

DECLARE

trainer\_specializare\_exception EXCEPTION;

cheie\_duplicata EXCEPTION;

---00001 CODUL ERORII PT ACEASTA EXCEPTIE

v\_verif NUMBER;

PRAGMA EXCEPTION\_INIT(cheie\_duplicata,-00001);

BEGIN

--INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

--VALUES (1, 1, 1);

--INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

--VALUES (2, 2, 3);

--INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

--VALUES (3, 3, 4);

--INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

--VALUES (4, 4, 2);

--INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

--VALUES (5, 5, 5);

--INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

--VALUES (5, 5, 5);

SELECT id\_specializare into v\_verif from specializare s where s.id\_specializare=200;

INSERT INTO trainer\_specializare (id\_trainer\_specializare, id\_trainer, id\_specializare)

VALUES (8, 4, 200);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

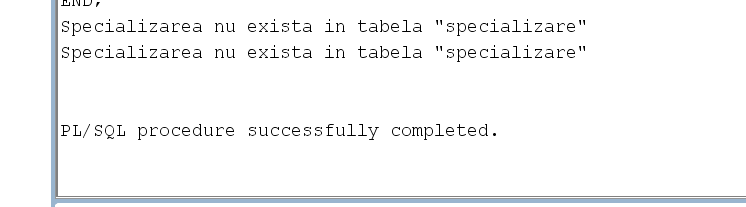
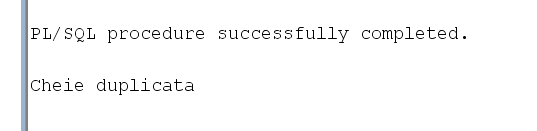
DBMS\_OUTPUT.PUT\_LINE('Specializarea nu exista in tabela "specializare"');

WHEN cheie\_duplicata THEN

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

END;

/



-- 4 Într-un bloc PL/SQL utilizati un cursor parametrizat pentru a prelua numele si adresa clientilor al caror abonament este dat ca parametru.

DECLARE

CURSOR client\_cursor (p\_id\_abonament IN NUMBER) IS

SELECT c.nume, c.adresa

FROM clienti c

WHERE c.id\_abonament = p\_id\_abonament;

v\_nume clienti.nume%TYPE;

v\_adresa clienti.adresa%TYPE;

id\_abonament\_cautat number:=&id\_abonament\_cautat;

verif number;

BEGIN

SELECT id\_tip\_abonament into verif from tip\_abonament a where id\_abonament\_cautat = a.id\_tip\_abonament;

FOR client\_c IN client\_cursor(id\_abonament\_cautat) LOOP

v\_nume := client\_c.nume;

v\_adresa := client\_c.adresa;

DBMS\_OUTPUT.PUT\_LINE('Nume: ' || v\_nume || ', Adresa: ' || v\_adresa);

END LOOP;

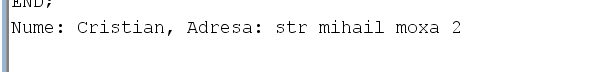
EXCEPTION

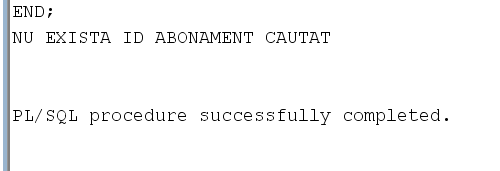
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA ID ABONAMENT CAUTAT');

END;

/





## **TEMA - Proiect SGBD, v.3**

Utilizând schema BD de la proiect (cel din semestrul 1, care va fi continuat în acest semestru), construiți cel puțin:

* 3 proceduri
* 3 funcții

(+ apelurile aferente)

*Notă*: se punctează complexitatea subprogramelor create; acestea ar trebui să includă simultan cât mai multe dintre mecanismele învățate: structuri de control, cursori, excepții, tipuri de date compuse...

-- 1. afisarea informatiilor despre tot clientii care au abonamente incheiate intr-un anumit interval de timp

CREATE OR REPLACE PROCEDURE afisare\_clienti\_abonamente

(data\_start IN DATE, data\_end IN DATE)

AS

v\_count INTEGER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM clienti c

JOIN informatii\_abonament\_client ia ON c.id\_abonament = ia.id\_abonament

JOIN tip\_abonament ta ON ia.id\_tip\_abonament = ta.id\_tip\_abonament

WHERE ia.data\_inceput BETWEEN data\_start AND data\_end;

IF v\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti in intervalul dorit!');

ELSE

FOR client IN (SELECT c.nume, c.prenume, c.adresa, ia.data\_inceput, ia.data\_sfarsit, ta.nume\_abonament, ta.pret, ta.descriere

FROM clienti c

JOIN informatii\_abonament\_client ia ON c.id\_abonament = ia.id\_abonament

JOIN tip\_abonament ta ON ia.id\_tip\_abonament = ta.id\_tip\_abonament

WHERE ia.data\_inceput BETWEEN data\_start AND data\_end)

LOOP

DBMS\_OUTPUT.PUT\_LINE(client.nume || ' ' || client.prenume || ', ' || client.adresa || ', Abonament ' || client.nume\_abonament || ', Pret: ' || client.pret || ' lei, Perioada: ' || client.data\_inceput || ' - ' || client.data\_sfarsit);

END LOOP;

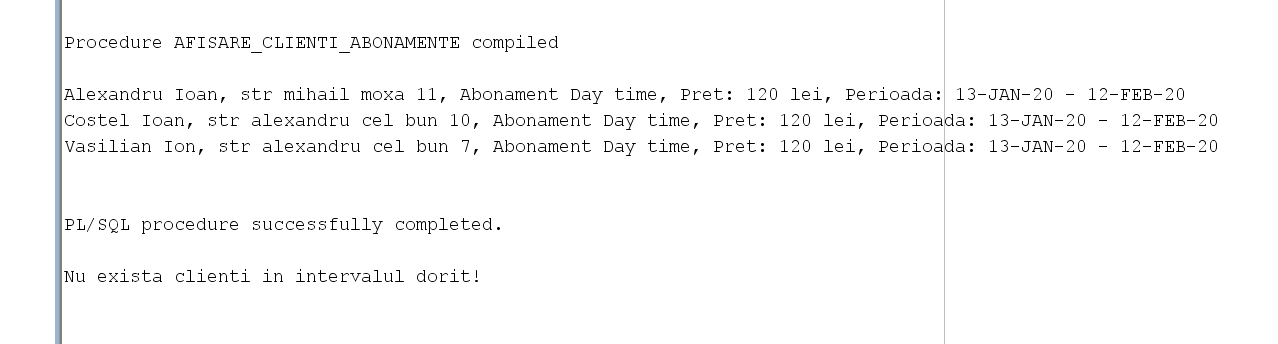
END IF;

END;

/

EXECUTE afisare\_clienti\_abonamente('01-JAN-2020', '31-JAN-2020');

EXECUTE afisare\_clienti\_abonamente('01-JAN-2023', '31-JAN-2023');



--2. Returnare numar de clienti cu abonament de tipul dat

CREATE OR REPLACE PROCEDURE nr\_clienti\_abonament

(p\_id\_abonament IN NUMBER, p\_nr\_clienti OUT NUMBER)

IS

BEGIN

SELECT COUNT(\*) INTO p\_nr\_clienti

FROM clienti

WHERE id\_abonament = p\_id\_abonament;

DBMS\_OUTPUT.PUT\_LINE('Numarul de clienti cu abonamentul cu id-ul '||p\_id\_abonament||' este: '||p\_nr\_clienti);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista abonament cu id-ul specificat');

END;

/

DECLARE

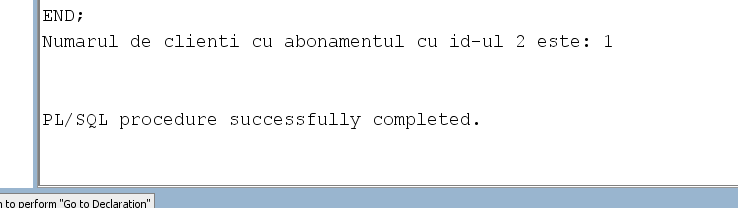
nr\_clienti NUMBER;

BEGIN

nr\_clienti\_abonament(&id\_abonament, nr\_clienti);

END;

/



-- 3. procedura - cel mai lung abonament

CREATE OR REPLACE PROCEDURE cel\_mai\_lung\_abonament

IS

v\_max NUMBER;

v\_id\_abonament informatii\_abonament\_client.id\_abonament%TYPE;

v\_data\_inceput informatii\_abonament\_client.data\_inceput%TYPE;

v\_data\_sfarsit informatii\_abonament\_client.data\_sfarsit%TYPE;

v\_diferenta NUMBER;

CURSOR c\_abonamente IS

SELECT id\_abonament, data\_inceput, data\_sfarsit

FROM informatii\_abonament\_client;

BEGIN

v\_max := 0;

v\_id\_abonament := NULL;

FOR abonament IN c\_abonamente LOOP

v\_diferenta := abonament.data\_sfarsit - abonament.data\_inceput;

IF v\_diferenta > v\_max THEN

v\_max := v\_diferenta;

v\_id\_abonament := abonament.id\_abonament;

v\_data\_inceput := abonament.data\_inceput;

v\_data\_sfarsit := abonament.data\_sfarsit;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Id-ul abonamentului cu cea mai lunga perioada este: ' || v\_id\_abonament);

DBMS\_OUTPUT.PUT\_LINE('Data inceput: ' || v\_data\_inceput);

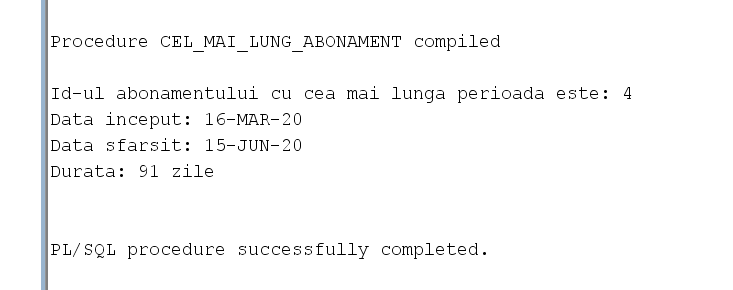
DBMS\_OUTPUT.PUT\_LINE('Data sfarsit: ' || v\_data\_sfarsit);

DBMS\_OUTPUT.PUT\_LINE('Durata: ' || v\_max || ' zile');

END;

/

EXECUTE cel\_mai\_lung\_abonament;



-- 1. numarul de clienti cu numele dat ca parametru

CREATE OR REPLACE FUNCTION numar\_clienti\_cu\_numele

(nume\_client IN VARCHAR2)

RETURN NUMBER

IS

nr\_clienti NUMBER := 0;

CURSOR c\_clienti IS SELECT COUNT(\*) AS nr\_clienti FROM clienti WHERE nume = nume\_client;

BEGIN

OPEN c\_clienti;

FETCH c\_clienti INTO nr\_clienti;

CLOSE c\_clienti;

DBMS\_OUTPUT.PUT\_LINE('Numarul de clienti cu numele ' || nume\_client || ' este: ' || nr\_clienti);

RETURN nr\_clienti;

END;

/

DECLARE

nr\_clienti NUMBER;

nr\_clienti2 NUMBER;

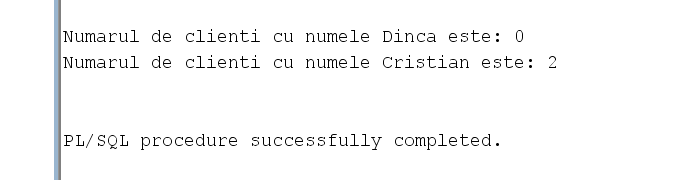
BEGIN

nr\_clienti := numar\_clienti\_cu\_numele('Dinca');

nr\_clienti2 := numar\_clienti\_cu\_numele('Cristian');

END;

/



-- 2. creeaza o functie care sa returneze un sir de caractere cu numele trainerului cu cele mai multe specializari, si numele lui

CREATE OR REPLACE PROCEDURE trainer\_max\_specializari1

IS

v\_nume\_trainer VARCHAR2(20);

v\_max\_specializari NUMBER(20);

BEGIN

SELECT t.nume || ' ' || t.prenume, COUNT(ts.id\_specializare) INTO v\_nume\_trainer, v\_max\_specializari

FROM traineri t

INNER JOIN trainer\_specializare ts ON t.id\_trainer = ts.id\_trainer

GROUP BY t.nume, t.prenume

HAVING COUNT(ts.id\_specializare) = (SELECT MAX(COUNT(id\_specializare)) FROM trainer\_specializare GROUP BY id\_trainer);

DBMS\_OUTPUT.PUT\_LINE('Trainerul cu cele mai multe specializari este: ' || v\_nume\_trainer || ' cu numarul de specializari: ' || v\_max\_specializari) ;

END;

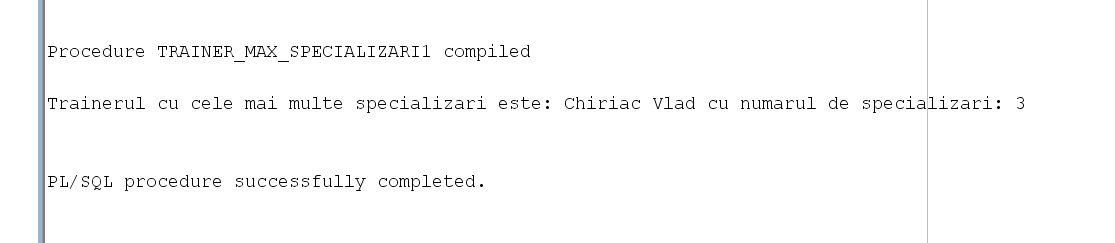
/

BEGIN

trainer\_max\_specializari1;

END;

/



-- 3. numarul de clase la care a participat clientul cu id-ul dat ca parametru

CREATE OR REPLACE FUNCTION numar\_clase\_client(p\_id\_client IN clienti.id\_client%TYPE)

RETURN NUMBER

IS

v\_nr\_clase NUMBER;

v\_id\_client number;

BEGIN

SELECT id\_client

into v\_id\_client

FROM clienti

WHERE p\_id\_client = id\_client;

SELECT COUNT(\*)

INTO v\_nr\_clase

FROM client\_programare cp

JOIN clienti c ON cp.id\_client = c.id\_client

JOIN informatii\_programare\_clasa ipc ON cp.id\_programare\_clasa = ipc.id\_programare\_clasa

WHERE c.id\_client = p\_id\_client;

DBMS\_OUTPUT.PUT\_LINE('Numarul de clase la care a participat clientul cu ID ' || p\_id\_client || ' este ' || v\_nr\_clase);

RETURN v\_nr\_clase;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista nicio programare la care a participat clientul cu ID ' || p\_id\_client);

RETURN NULL;

END;

/

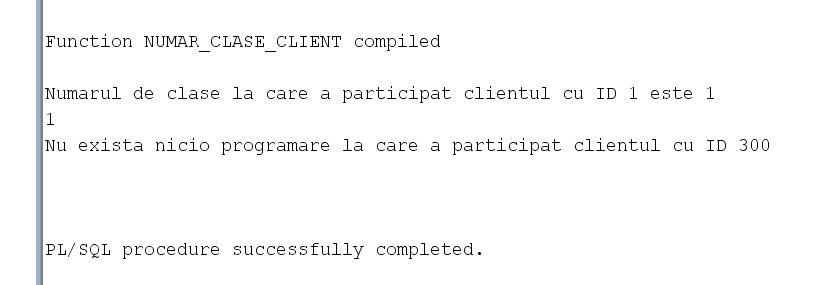
BEGIN

DBMS\_OUTPUT.PUT\_LINE(numar\_clase\_client(1));

DBMS\_OUTPUT.PUT\_LINE(numar\_clase\_client(300));

END;

/



**Pachete**

CREATE OR REPLACE PACKAGE pachet\_subprograme1 IS

PROCEDURE afisare\_clienti\_abonamente(data\_start IN DATE, data\_end IN DATE);

PROCEDURE nr\_clienti\_abonament(p\_id\_abonament IN NUMBER, p\_nr\_clienti OUT NUMBER);

PROCEDURE cel\_mai\_lung\_abonament;

END pachet\_subprograme1;

/

CREATE OR REPLACE PACKAGE BODY pachet\_subprograme1 IS

PROCEDURE afisare\_clienti\_abonamente(data\_start IN DATE, data\_end IN DATE)

AS

v\_count INTEGER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM clienti c

JOIN informatii\_abonament\_client ia ON c.id\_abonament = ia.id\_abonament

JOIN tip\_abonament ta ON ia.id\_tip\_abonament = ta.id\_tip\_abonament

WHERE ia.data\_inceput BETWEEN data\_start AND data\_end;

IF v\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti in intervalul dorit!');

ELSE

FOR client IN (SELECT c.nume, c.prenume, c.adresa, ia.data\_inceput, ia.data\_sfarsit, ta.nume\_abonament, ta.pret, ta.descriere

FROM clienti c

JOIN informatii\_abonament\_client ia ON c.id\_abonament = ia.id\_abonament

JOIN tip\_abonament ta ON ia.id\_tip\_abonament = ta.id\_tip\_abonament

WHERE ia.data\_inceput BETWEEN data\_start AND data\_end)

LOOP

DBMS\_OUTPUT.PUT\_LINE(client.nume || ' ' || client.prenume || ', ' || client.adresa || ', Abonament ' || client.nume\_abonament || ', Pret: ' || client.pret || ' lei, Perioada: ' || client.data\_inceput || ' - ' || client.data\_sfarsit);

END LOOP;

END IF;

END;

PROCEDURE nr\_clienti\_abonament(p\_id\_abonament IN NUMBER, p\_nr\_clienti OUT NUMBER)

IS

BEGIN

SELECT COUNT(\*)

INTO p\_nr\_clienti

FROM clienti

WHERE id\_abonament = p\_id\_abonament;

DBMS\_OUTPUT.PUT\_LINE('Numarul de clienti cu abonamentul cu id-ul '||p\_id\_abonament||' este: '||p\_nr\_clienti);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista abonament cu id-ul specificat');

END;

PROCEDURE cel\_mai\_lung\_abonament

IS

v\_max NUMBER;

v\_id\_abonament informatii\_abonament\_client.id\_abonament%TYPE;

v\_data\_inceput informatii\_abonament\_client.data\_inceput%TYPE;

v\_data\_sfarsit informatii\_abonament\_client.data\_sfarsit%TYPE;

v\_diferenta NUMBER;

CURSOR c\_abonamente IS

SELECT id\_abonament, data\_inceput, data\_sfarsit

FROM informatii\_abonament\_client;

BEGIN

v\_max := 0;

v\_id\_abonament := NULL;

FOR abonament IN c\_abonamente LOOP

v\_diferenta := abonament.data\_sfarsit - abonament.data\_inceput;

IF v\_diferenta > v\_max THEN

v\_max := v\_diferenta;

v\_id\_abonament := abonament.id\_abonament;

v\_data\_inceput := abonament.data\_inceput;

v\_data\_sfarsit := abonament.data\_sfarsit;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Id-ul abonamentului cu cea mai lunga perioada este: ' || v\_id\_abonament);

DBMS\_OUTPUT.PUT\_LINE('Data inceput: ' || v\_data\_inceput);

DBMS\_OUTPUT.PUT\_LINE('Data sfarsit: ' || v\_data\_sfarsit);

DBMS\_OUTPUT.PUT\_LINE('Durata: ' || v\_max || ' zile');

END;

END pachet\_subprograme1;

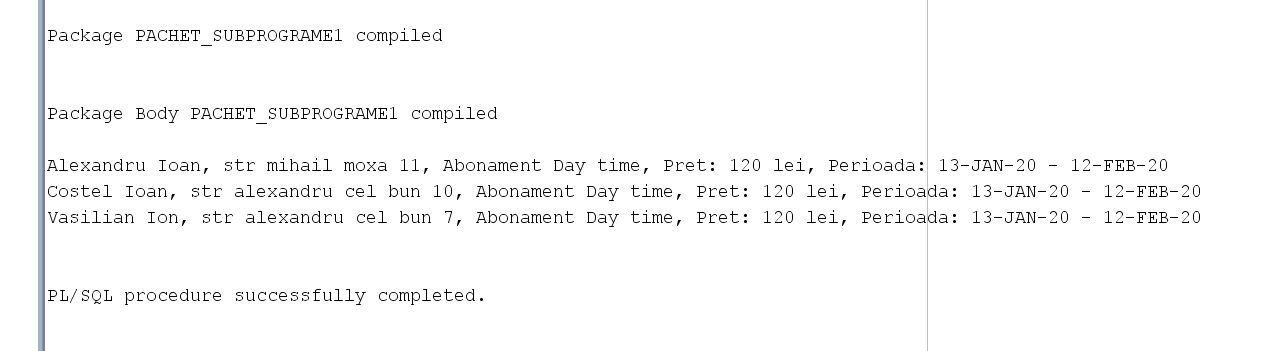
/

BEGIN

pachet\_subprograme1.afisare\_clienti\_abonamente('01-JAN-2020', '31-JAN-2020');

END;

/



**TRIGERI**

CREATE OR REPLACE TRIGGER trg\_update\_tip\_ab\_cascada

BEFORE UPDATE OF id\_tip\_abonament ON tip\_abonament

FOR EACH ROW

DECLARE

affected\_rows NUMBER;

BEGIN

UPDATE informatii\_abonament\_client

SET id\_tip\_abonament = :NEW.id\_tip\_abonament

WHERE id\_tip\_abonament = :OLD.id\_tip\_abonament;

END;

/

UPDATE tip\_abonament

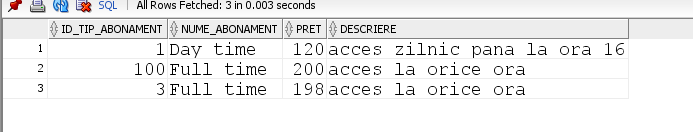
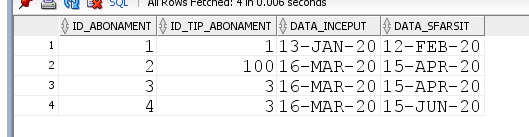
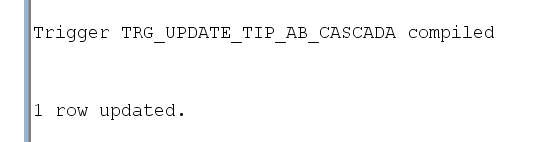
SET id\_tip\_abonament=100

WHERE id\_tip\_abonament=2;

/

select \* from informatii\_abonament\_client;

select \* from tip\_abonament;



--2. -- sa se creeze un declansator prin care sa se impiedice adaugarea unei noi

-- specializari pentru un trainer daca are deja 3 specializari

CREATE OR REPLACE TRIGGER trg\_check\_trainer\_specializare

BEFORE INSERT ON trainer\_specializare

FOR EACH ROW

DECLARE

v\_trainer\_id traineri.id\_trainer%TYPE;

v\_specializari\_count NUMBER;

BEGIN

-- Obținem id-ul trainerului pentru care se adaugă specializarea

SELECT id\_trainer INTO v\_trainer\_id FROM traineri WHERE id\_trainer = :NEW.id\_trainer\_specializare;

-- Numărăm specializările existente pentru trainer

SELECT COUNT(\*) INTO v\_specializari\_count FROM trainer\_specializare WHERE id\_trainer = v\_trainer\_id;

-- Verificăm dacă numărul de specializări este deja 3

IF v\_specializari\_count >= 3 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Trainerul are deja 3 specializări. Nu se poate adăuga o nouă specializare.');

END IF;

END;

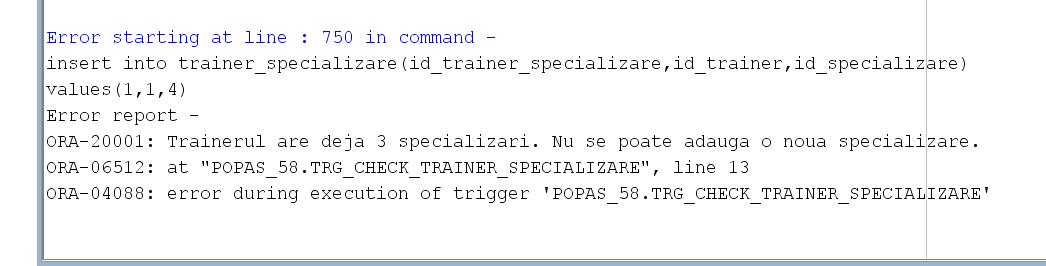
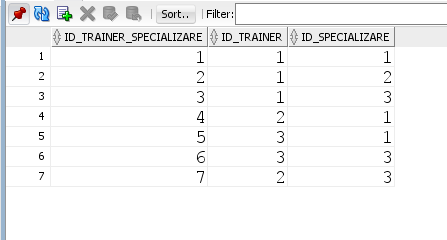
/

insert into specializare(id\_specializare, nume\_specializare, descriere)

values(4,'box','Sport in care doi adversari lupta între ei, pe ring, după anumite reguli, cu pumnii îmbracati in manusi speciale.');

insert into trainer\_specializare(id\_trainer\_specializare,id\_trainer,id\_specializare)

values(1,1,4);



-- 3. sa se verifice daca sala este full cand se adauga un client la programare

CREATE OR REPLACE TRIGGER trg\_adaugare\_programari

BEFORE INSERT ON client\_programare

FOR EACH ROW

DECLARE

numar\_clienti NUMBER;

dimensiune\_sala NUMBER;

BEGIN

SELECT COUNT(\*) INTO numar\_clienti

FROM client\_programare

WHERE id\_programare\_clasa = :NEW.id\_programare\_clasa;

SELECT dimensiune INTO dimensiune\_sala

FROM sala

WHERE id\_sala = (SELECT id\_sala FROM informatii\_programare\_clasa WHERE id\_programare\_clasa = :NEW.id\_programare\_clasa);

IF numar\_clienti > dimensiune\_sala THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Sala este ocupată la maxim.');

END IF;

END;

/

insert into sala(id\_sala, dimensiune)

values(5,1);

insert into informatii\_programare\_clasa(id\_programare\_clasa, id\_specializare, id\_sala, data\_zi, ora\_inceput, ora\_sfarsit)

values(100,3,5,to\_date('28/01/2023', 'DD/MM/YYYY'),to\_date('15:30', 'HH24:MI'),to\_date('17:30', 'HH24:MI'));

insert into client\_programare(id\_client\_programare,id\_client,id\_programare\_clasa)

values(800,1,100);

insert into client\_programare(id\_client\_programare,id\_client,id\_programare\_clasa)

values(801,2,100);