Gestiunea

unei

Aplicatii de fitness

Popescu Mihaela-Maria

An II – Grupa 241

1. ***Prezentați pe scurt baza de date (utilitatea ei).***

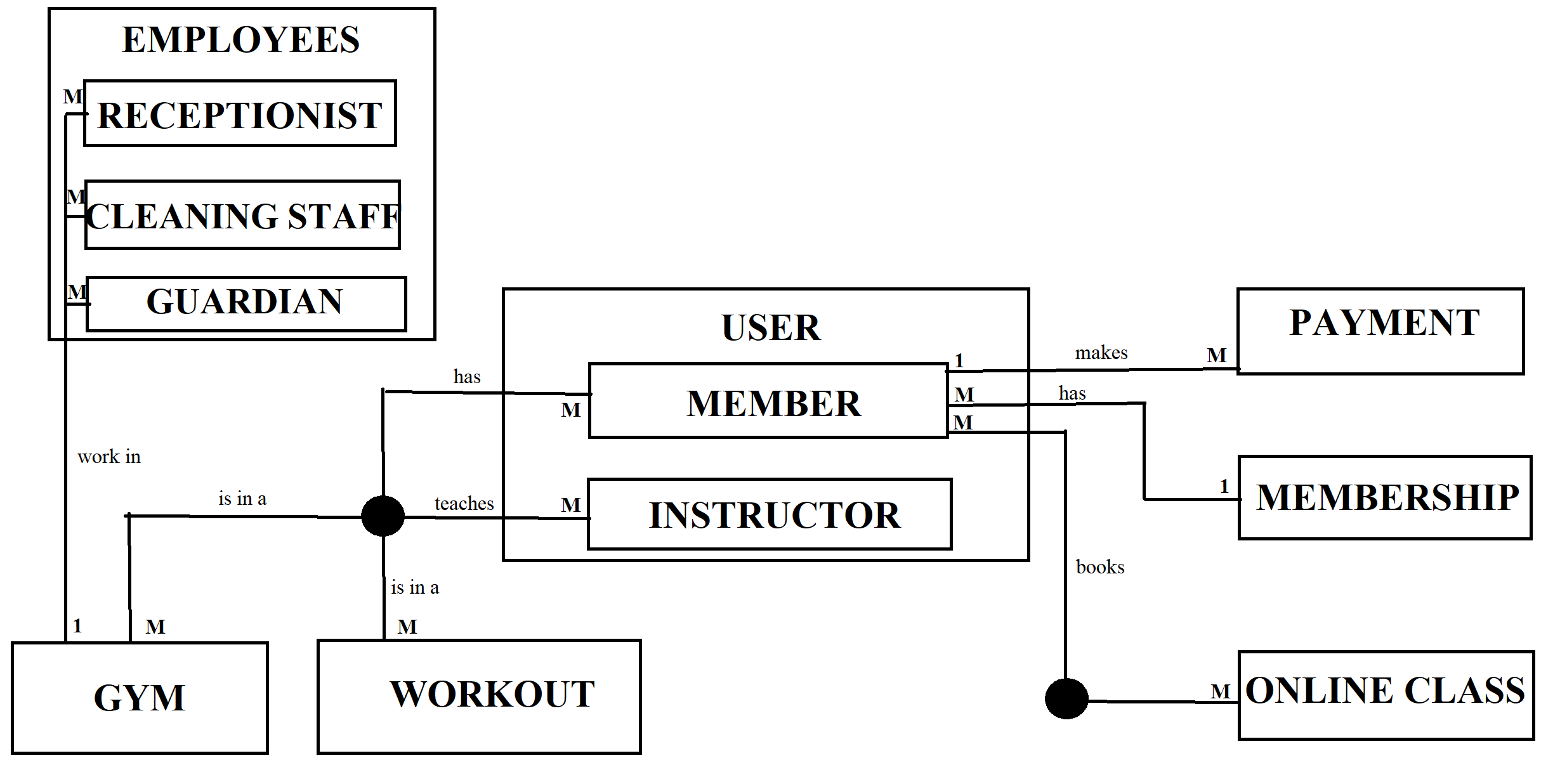
In scopul realizarii proiectului final la materia Sisteme de Gestiune a Bazelor de Date, am construit baza de date a unei aplicatii de fitness.

In cadrul acesteia retinem informatii despre toti userii aplicatiei, atat membrii cat si instructorii salilor. Membrii detin un anumit tip de abonament, carora li se atribuie un program de antrenament cu un instructor personal la unul dintre cluburi. Fiecare antrenament este format din mai multe exercitii. Toti membrii achita o taxa lunara, tinandu-se evidenta platilor. Instructorul actualizeaza prin intermediul aplicatiei antrenamentele membrilor. Un membru poate avea mai multe tipuri de antrenamente, cu diferiti instructori si la diferite sali. Exista optiunea de antrenamente online, astfel un membru poate sa achizitioneze un antrenament online.

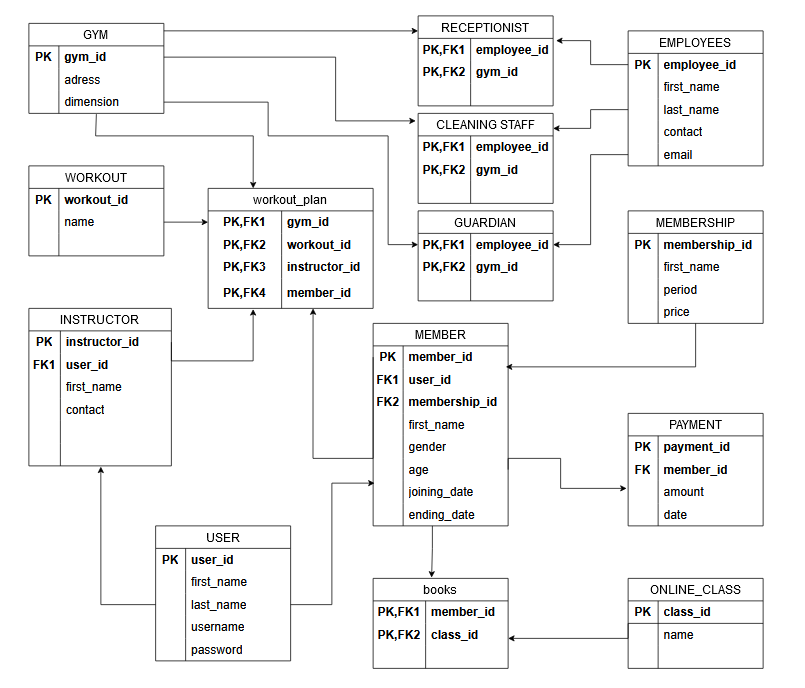
De asemenea, retinem date si despre angajatii fiecarei sali. Orice sala poate avea unul sau mai multi receptionisti, paznici sau personal de curatenie, insa fiecare angajat poate lucra la o singura sala.

Pentru o organizare eficienta, am utilizat doua tabele asociative ce ne ajuta in gestionarea relatiilor many-to-many: “workout\_plan” ce face legatura intre 4 entitati(Gym, Instructor, Workout, Member) si “books” intre Member si Online\_Class.

1. ***Realizați diagrama entitate-relație(ERD).***



1. ***Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.***



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

CREATE TABLE GYM

(GYM\_ID NUMBER(10) PRIMARY KEY, ADRESS VARCHAR(20), DIMENSION NUMBER(10));

CREATE TABLE WORKOUT

(WORKOUT\_ID NUMBER(10) PRIMARY KEY, NAME VARCHAR(20) NOT NULL);

CREATE TABLE USERS

(USER\_ID NUMBER(10) PRIMARY KEY, FIRST\_NAME VARCHAR(20) NOT NULL, LAST\_NAME VARCHAR(20) NOT NULL,USERNAME VARCHAR(20), PASSWORD VARCHAR(20) NOT NULL);

CREATE TABLE ONLINE\_CLASS

(CLASS\_ID NUMBER(10) PRIMARY KEY , NAME VARCHAR(20) NOT NULL);

CREATE TABLE MEMBERSHIP

(MEMBERSHIP\_ID NUMBER(10) PRIMARY KEY , NAME VARCHAR(20) NOT NULL, PERIOD VARCHAR(20)NOT NULL, PRICE NUMBER(10)NOT NULL);

CREATE TABLE MEMBERS

(MEMBER\_ID NUMBER(10) PRIMARY KEY , FIRST\_NAME VARCHAR(20), GENDER VARCHAR(20), AGE NUMBER(10), JOINING\_DATE DATE NOT NULL, ENDING\_DATE DATE,

MEMBERSHIP\_ID REFERENCES MEMBERSHIP(MEMBERSHIP\_ID), USER\_ID REFERENCES USER1(USER\_ID));

CREATE TABLE INSTRUCTOR

(INSTRUCTOR\_ID NUMBER(10) PRIMARY KEY, FIRST\_NAME VARCHAR(20) NOT NULL, CONTACT VARCHAR(20), USER\_ID REFERENCES USER1(USER\_ID));

CREATE TABLE PAYMENT

(PAYMENT\_ID NUMBER(10) PRIMARY KEY, AMOUNT NUMBER(10) NOT NULL,DATE\_TIME DATE, MEMBER\_ID REFERENCES MEMBER1(MEMBER\_ID));

CREATE TABLE WORKOUT\_PLAN

(GYM\_ID NUMBER(10) REFERENCES GYM(GYM\_ID), WORKOUT\_ID NUMBER(10) REFERENCES WORKOUT(WORKOUT\_ID),INSTRUCTOR\_ID NUMBER(10) REFERENCES INSTRUCTOR(INSTRUCTOR\_ID),MEMBER\_ID NUMBER(10) REFERENCES MEMBER1(MEMBER\_ID),

PRIMARY KEY(GYM\_ID,WORKOUT\_ID,INSTRUCTOR\_ID,MEMBER\_ID));

CREATE TABLE BOOKS

(MEMBER\_ID NUMBER(10) REFERENCES MEMBERS(MEMBER\_ID), CLASS\_ID NUMBER(10) REFERENCES ONLINE\_CLASS(CLASS\_ID),

PRIMARY KEY(MEMBER\_ID,CLASS\_ID));

CREATE TABLE EMPLOYEES

(EMPLOYEE\_ID NUMBER(10) PRIMARY KEY , FIRST\_NAME VARCHAR(20) NOT NULL, LAST\_NAME VARCHAR(20) NOT NULL,CONTACT VARCHAR(20), EMAIL VARCHAR(20));

CREATE TABLE RECEPTIONISTS

(EMPLOYEE\_ID NUMBER(10) REFERENCES EMPLOYEES(EMPLOYEE\_ID), GYM\_ID NUMBER(10) REFERENCES GYMS(GYM\_ID),

PRIMARY KEY (EMPLOYEE\_ID,GYM\_ID));

CREATE TABLE CLEANING\_STAFF

(EMPLOYEE\_ID NUMBER(10) REFERENCES EMPLOYEES(EMPLOYEE\_ID), GYM\_ID NUMBER(10) REFERENCES GYMS(GYM\_ID),

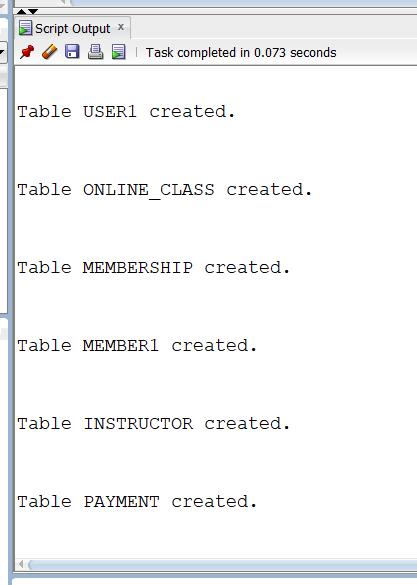
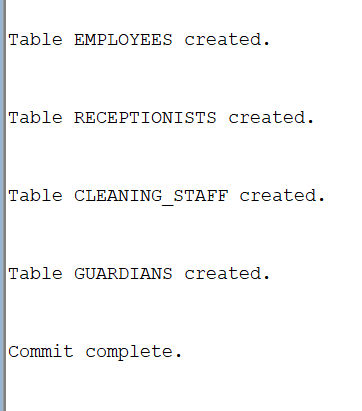
PRIMARY KEY (EMPLOYEE\_ID,GYM\_ID));

CREATE TABLE GUARDIANS

(EMPLOYEE\_ID NUMBER(10) REFERENCES EMPLOYEES(EMPLOYEE\_ID), GYM\_ID NUMBER(10) REFERENCES GYMS(GYM\_ID),

PRIMARY KEY (EMPLOYEE\_ID,GYM\_ID));

COMMIT;

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

*TABELA GYMS:*

INSERT INTO GYMS

VALUES (1, 'STR SOLDAT IOSIF ION', 10);

INSERT INTO GYMS

VALUES (2, 'STR ISTRIEI', 10);

INSERT INTO GYMS

VALUES (3, 'STR LUNCA', 100);

INSERT INTO GYMS

VALUES (4, 'STR LALOSU', 80);

INSERT INTO GYMS

VALUES (5, 'STR PADESU', 120);

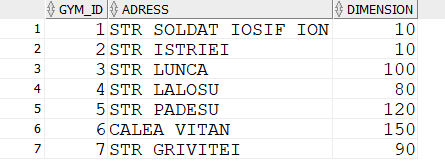
INSERT INTO GYMS

VALUES (6, 'CALEA VITAN', 150);

INSERT INTO GYMS

VALUES (7, 'STR GRIVITEI', 90);

SELECT \* FROM GYMS;



*--TABELA WORKOUT*

INSERT INTO WORKOUT

VALUES (1, 'CARDIO');

INSERT INTO WORKOUT

VALUES (2, 'CROSS-FIT');

INSERT INTO WORKOUT

VALUES (3, 'SWIMMING');

INSERT INTO WORKOUT

VALUES (4, 'POWERLIFTING');

INSERT INTO WORKOUT

VALUES (5, 'BODYBUILDING');

INSERT INTO WORKOUT

VALUES (6, 'ZUMBA');

INSERT INTO WORKOUT

VALUES (7, 'STRENGTH TRAINING');

INSERT INTO WORKOUT

VALUES (8, 'CALISTHENICS');

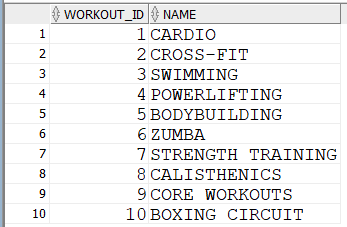
INSERT INTO WORKOUT

VALUES (9, 'CORE WORKOUTS');

INSERT INTO WORKOUT

VALUES (10, 'BOXING CIRCUIT');

SELECT \* FROM WORKOUT;



*--TABELA EMPLOYEES*

INSERT INTO EMPLOYEES

VALUES (100, 'MARIA','GEORGESCU','0722513230','MARIA@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (101, 'IOANA','TRANDAFIR','0722518830','IOANA@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (102, 'CORNEL','PROTOPOPESCU','0729513230','POPESCU@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (103, 'ALEX','MIHAIL','0722763230','ALEX@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (104, 'IUSTINA','MARGINEANU','072876530','MARGINEANU@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (105, 'ROBERT','IONESCU','0722777230','ROB@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (106, 'DAVID','BRANZOVENESCU','0733313230','BRANZICA@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (107, 'IULIANA','POPESCU','078899550','POPESCU32@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (108, 'CRISTIAN','IOAN','0722444230','IOAN@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (109, 'GABRIEL','BADEA','0797613230','GB@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (110, 'AURORA','IANCU','0722777290','IANCU\_A@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (111, 'BOGDAN','DELAP','079088765','DLP@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (112, 'KARINA','LUCULET','074443230','KARINA@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (113, 'VALENTIN','ALAFIN','0711113230','VFIN@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (114, 'DRAGOS','CONSTANTIN','079123430','CONSTANTIN@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (115, 'CASSANDRA','MARTIN','071556730','CASSI@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (116, 'GEORGIANA','ANDU','0717789230','GEORGIA@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (117, 'RALUCA','LEON','0745564330','RALEO@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (118, 'MARIANA','ANGHEL','0785497860','ANGHEL@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (119, 'BIANCA','OANCEA','079875550','BIBI@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (120, 'DANA','EPU','0776865231','EPU\_DI@GMAIL.COM');

INSERT INTO EMPLOYEES

VALUES (121, 'MARA','MOCANU','0722213230','MARA@GMAIL.COM');

SELECT \* FROM EMPLOYEES;



*--TABELA RECEPTIONISTS*

INSERT INTO RECEPTIONISTS

VALUES(100,1);

INSERT INTO RECEPTIONISTS

VALUES(101,2);

INSERT INTO RECEPTIONISTS

VALUES(102,3);

INSERT INTO RECEPTIONISTS

VALUES(103,4);

INSERT INTO RECEPTIONISTS

VALUES(104,5);

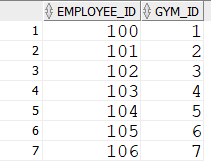
INSERT INTO RECEPTIONISTS

VALUES(105,6);

INSERT INTO RECEPTIONISTS

VALUES(106,7);

SELECT \* FROM RECEPTIONISTS;



*--TABELA GUARDIANS*

INSERT INTO GUARDIANS

VALUES(107,1);

INSERT INTO GUARDIANS

VALUES(108,2);

INSERT INTO GUARDIANS

VALUES(109,3);

INSERT INTO GUARDIANS

VALUES(110,4);

INSERT INTO GUARDIANS

VALUES(111,5);

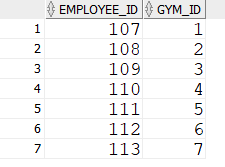
INSERT INTO GUARDIANS

VALUES(112,6);

INSERT INTO GUARDIANS

VALUES(113,7);

SELECT \* FROM GUARDIANS;



*--TABELA CLEANING\_STAFF*

INSERT INTO CLEANING\_STAFF

VALUES(114,1);

INSERT INTO CLEANING\_STAFF

VALUES(115,2);

INSERT INTO CLEANING\_STAFF

VALUES(116,3);

INSERT INTO CLEANING\_STAFF

VALUES(117,4);

INSERT INTO CLEANING\_STAFF

VALUES(118,5);

INSERT INTO CLEANING\_STAFF

VALUES(119,6);

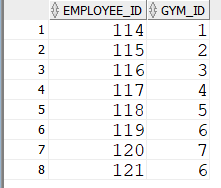
INSERT INTO CLEANING\_STAFF

VALUES(120,7);

INSERT INTO CLEANING\_STAFF

VALUES(121,6);

SELECT \* FROM CLEANING\_STAFF;



*--TABELA USERS*

INSERT INTO USERS

VALUES (1, 'MIHAELA','POPESCU','mica','cefef!2dA');

INSERT INTO USERS

VALUES (2, 'DIANA','IORDACHE','didi\_iord','jncUBB');

INSERT INTO USERS

VALUES (3, 'MARIA','MARIN','MM36','LKLjh');

INSERT INTO USERS

VALUES (4, 'COSMIN','TEODOR','cosmit','kfkw!233dA');

INSERT INTO USERS

VALUES (5, 'ALEXANDRU','RUS','777\_MONEY','lajdeb@AX');

INSERT INTO USERS

VALUES (6, 'MATEI','RUS','forex\_mr','mxrcars\*');

INSERT INTO USERS

VALUES (7, 'CORINA','NITULESCU','cory','asdd!sgbd');

INSERT INTO USERS

VALUES (8, 'NATALIA','IONESCU','nationescu','mvsjhd!#@');

INSERT INTO USERS

VALUES (9, 'EUGEN','POPESCU','popyeug','ddvvw22');

INSERT INTO USERS

VALUES (10, 'LILIANA','LUCULET','lilyluk','fb!fe31w3');

INSERT INTO USERS

VALUES (11, 'RADU','RUS','valer\_rus','cnjskme');

INSERT INTO USERS

VALUES (12, 'MIRELA','RUS','t\_mirela','vgdbrgr');

INSERT INTO USERS

VALUES (13, 'IONEL','OCO','oko\_ion','wq32213');

INSERT INTO USERS

VALUES (14, 'ALEXANDRA','GHEORGHE','alxg','wr3r3f');

INSERT INTO USERS

VALUES (15, 'MARA','MALUTAN','maruca','sdrg222');

INSERT INTO USERS

VALUES (16, 'PATRICIA','TRANDAFIR','floricica','erfcwv332');

INSERT INTO USERS

VALUES (17, 'MARIA','DUMITRU','mariuca','fvrgrwf');

INSERT INTO USERS

VALUES (18, 'VLAD','PARASCHIV','vllladd','asdefe');

INSERT INTO USERS

VALUES (19, 'LUCA','DITA','luked','ceffkjwefn');

INSERT INTO USERS

VALUES (20, 'CLAUDIA','ERIN','clauclau','vwefqe');

INSERT INTO USERS

VALUES (21, 'TUDOR','OPRISAN','tud\_oop','ecdvreg3');

INSERT INTO USERS

VALUES (22, 'SERGIU','ARIN','sssergiu','lmnbhj');

INSERT INTO USERS

VALUES (23, 'ADRIAN','ZAMIN','ADIZZZ','ffvdaqqq');

INSERT INTO USERS

VALUES (24, 'LAURENTIU','FLORENTIN','florentiu','xscvvvds');

INSERT INTO USERS

VALUES (25, 'LORENA','NEAGU','lolo\_nnn','lolkcnjsd');

INSERT INTO USERS

VALUES (26, 'GABRIELA','GLONT','GG\_WP','dnkjsndjn');

INSERT INTO USERS

VALUES (27, 'RAUL','GHEORGHIU','raul\_gs','DFwdcf');

INSERT INTO USERS

VALUES (28, 'FLORIN','MIHAI','florinmihai','cofkiebf');

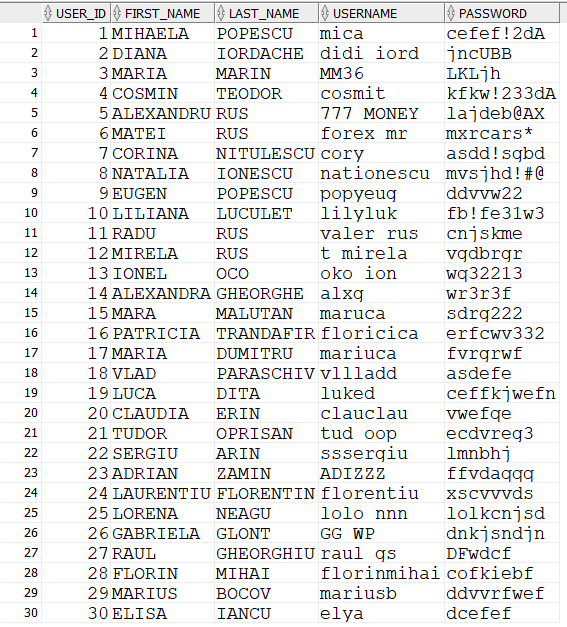
INSERT INTO USERS

VALUES (29, 'MARIUS','BOCOV','mariusb','ddvvrfwef');

INSERT INTO USERS

VALUES (30, 'ELISA','IANCU','elya','dcefef');

select \* from users;



*--TABELA INSTRUCTORS*

INSERT INTO INSTRUCTORS

VALUES(1,'MIHAELA','072662512',1);

INSERT INTO INSTRUCTORS

VALUES(2,'ALEX','073451512',5);

INSERT INTO INSTRUCTORS

VALUES(3,'MATEI','077876251',6);

INSERT INTO INSTRUCTORS

VALUES(4,'NATY','071982611',8);

INSERT INTO INSTRUCTORS

VALUES(5,'FLO','07578912',29);

INSERT INTO INSTRUCTORS

VALUES(6,'SERGIU','079875123',22);

INSERT INTO INSTRUCTORS

VALUES(7,'GABI','0789007611',26);

INSERT INTO INSTRUCTORS

VALUES(8,'PATY','079876112',16);

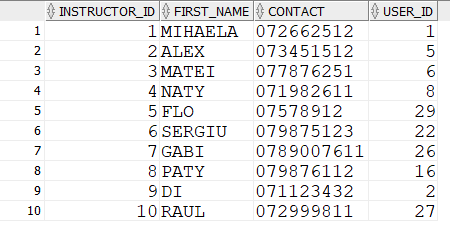
INSERT INTO INSTRUCTORS

VALUES(9,'DI','071123432',2);

INSERT INTO INSTRUCTORS

VALUES(10,'RAUL','072999811',27);

select \* from instructors;



*--TABLE ONLINE\_CLASS*

INSERT INTO ONLINE\_CLASS

VALUES(1001,'CYCLING');

INSERT INTO ONLINE\_CLASS

VALUES(1002,'TRAIN ABS!');

INSERT INTO ONLINE\_CLASS

VALUES(1003,'PILATES');

INSERT INTO ONLINE\_CLASS

VALUES(1004,'YOGA');

INSERT INTO ONLINE\_CLASS

VALUES(1005,'DUMP TRUCK');

INSERT INTO ONLINE\_CLASS

VALUES(1006,'DANCE IT OUT');

INSERT INTO ONLINE\_CLASS

VALUES(1007,'SWEATY SWEAT');

INSERT INTO ONLINE\_CLASS

VALUES(1008,'MUSCLE DESTROYER');

INSERT INTO ONLINE\_CLASS

VALUES(1009,'FULL BODY TONING');

INSERT INTO ONLINE\_CLASS

VALUES(1010,'BEGINNERS CIRCUIT');

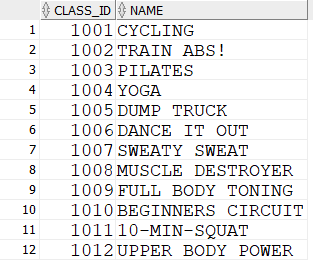
INSERT INTO ONLINE\_CLASS

VALUES(1011,'10-MIN-SQUAT');

INSERT INTO ONLINE\_CLASS

VALUES(1012,'UPPER BODY POWER');

SELECT \* FROM ONLINE\_CLASS;



*--TABELA MEMBERSHIP*

INSERT INTO MEMBERSHIP

VALUES(1,'BRONZE','1 MONTH',199);

INSERT INTO MEMBERSHIP

VALUES(2,'SILVER','1 MONTH',239);

INSERT INTO MEMBERSHIP

VALUES(3,'GOLD','1 MONTH',279);

INSERT INTO MEMBERSHIP

VALUES(4,'PLATINUM','1 MONTH',299);

INSERT INTO MEMBERSHIP

VALUES(5,'BRONZE','3 MONTHS',559);

INSERT INTO MEMBERSHIP

VALUES(6,'SILVER','3 MONTHS',699);

INSERT INTO MEMBERSHIP

VALUES(7,'GOLD','3 MONTH',799);

INSERT INTO MEMBERSHIP

VALUES(8,'PLATINUM','3 MONTH',899);

INSERT INTO MEMBERSHIP

VALUES(9,'BRONZE','12 MONTHS',1990);

INSERT INTO MEMBERSHIP

VALUES(10,'SILVER','12 MONTHS',2590);

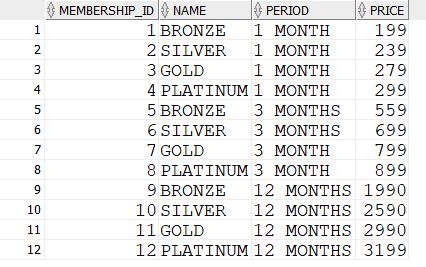
INSERT INTO MEMBERSHIP

VALUES(11,'GOLD','12 MONTHS',2990);

INSERT INTO MEMBERSHIP

VALUES(12,'PLATINUM','12 MONTHS',3199);

SELECT \* FROM MEMBERSHIP;



--TABELA MEMBERS

INSERT INTO MEMBERS

VALUES(2001,'MARIA','F',19,SYSDATE,DATE '2023-02-12',1,3);

INSERT INTO MEMBERS

VALUES(2002,'COSMIN','M',23,SYSDATE,DATE'2023-04-12',5,4);

INSERT INTO MEMBERS

VALUES(2003,'CORINA','F',25,SYSDATE,DATE'2023-02-12',3,7);

INSERT INTO MEMBERS

VALUES(2004,'EUGEN','M',54,SYSDATE,DATE'2023-02-12',1,9);

INSERT INTO MEMBERS

VALUES(2005,'LILIANA','F',55,SYSDATE,DATE'2024-01-12',12,10);

INSERT INTO MEMBERS

VALUES(2006,'RADU','M',48,SYSDATE,DATE'2023-02-12',3,11);

INSERT INTO MEMBERS

VALUES(2007,'MIRELA','F',49,SYSDATE,DATE'2023-04-12',7,12);

INSERT INTO MEMBERS

VALUES(2008,'IONEL','M',40,SYSDATE,DATE'2024-01-12',9,13);

INSERT INTO MEMBERS

VALUES(2009,'ELISA','F',19,SYSDATE,DATE'2023-04-12',6,30);

INSERT INTO MEMBERS

VALUES(2010,'FLORIN','M',22,SYSDATE,DATE'2023-02-12',1,28);

INSERT INTO MEMBERS

VALUES(2011,'LORENA','F',17,SYSDATE,DATE'2023-02-12',2,25);

INSERT INTO MEMBERS

VALUES(2012,'LAURENTIU','M',21,SYSDATE,DATE'2023-02-12',3,24);

INSERT INTO MEMBERS

VALUES(2013,'ALEXANDRA','F',18,SYSDATE,DATE'2023-02-12',1,14);

INSERT INTO MEMBERS

VALUES(2014,'MARIA','F',24,SYSDATE,DATE'2023-02-12',4,17);

INSERT INTO MEMBERS

VALUES(2015,'TUDOR','M',30,SYSDATE,DATE '2023-02-12',3,21);

INSERT INTO MEMBERS

VALUES(2016,'ADRIAN','M',39,SYSDATE,DATE'2023-04-12',6,23);

INSERT INTO MEMBERS

VALUES(2017,'MARA','F',27,SYSDATE,DATE'2023-02-12',1,15);

INSERT INTO MEMBERS

VALUES(2018,'VLAD','M',29,SYSDATE,DATE'2024-01-12',11,18);

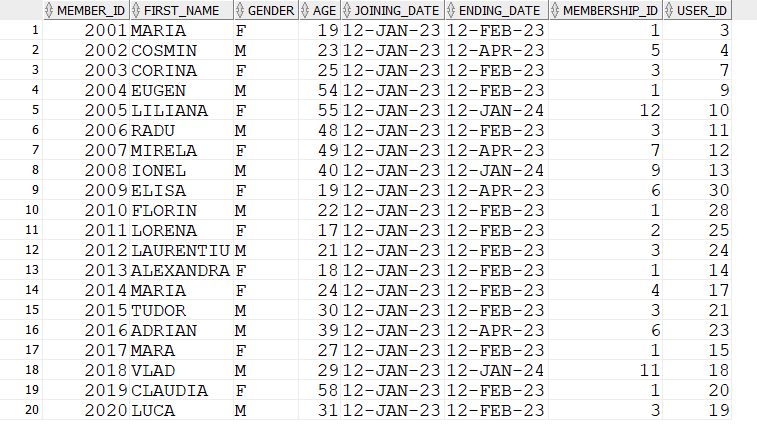
INSERT INTO MEMBERS

VALUES(2019,'CLAUDIA','F',58,SYSDATE,DATE'2023-02-12',1,20);

INSERT INTO MEMBERS

VALUES(2020,'LUCA','M',31,SYSDATE,DATE '2023-02-12',3,19);

SELECT \* FROM MEMBERS;



--TABELA ASOCIATIVA WORKOUT\_PLAN

INSERT INTO WORKOUT\_PLAN

VALUES(1,9,10,2020);

INSERT INTO WORKOUT\_PLAN

VALUES(2,10,3,2002);

INSERT INTO WORKOUT\_PLAN

VALUES(3,2,4,2007);

INSERT INTO WORKOUT\_PLAN

VALUES(1,6,2,2005);

INSERT INTO WORKOUT\_PLAN

VALUES(1,7,2,2005);

INSERT INTO WORKOUT\_PLAN

VALUES(1,3,2,2005);

INSERT INTO WORKOUT\_PLAN

VALUES(5,4,6,2015);

INSERT INTO WORKOUT\_PLAN

VALUES(7,3,8,2009);

INSERT INTO WORKOUT\_PLAN

VALUES(7,5,5,2018);

INSERT INTO WORKOUT\_PLAN

VALUES(6,1,1,2006);

INSERT INTO WORKOUT\_PLAN

VALUES(6,1,1,2007);

INSERT INTO WORKOUT\_PLAN

VALUES(6,1,2,2008);

INSERT INTO WORKOUT\_PLAN

VALUES(6,1,2,2005);

INSERT INTO WORKOUT\_PLAN

VALUES(6,3,2,2004);

INSERT INTO WORKOUT\_PLAN

VALUES(2,7,3,2013);

INSERT INTO WORKOUT\_PLAN

VALUES(7,5,4,2016);

INSERT INTO WORKOUT\_PLAN

VALUES(4,1,1,2011);

INSERT INTO WORKOUT\_PLAN

VALUES(4,2,1,2011);

INSERT INTO WORKOUT\_PLAN

VALUES(4,3,1,2011);

INSERT INTO WORKOUT\_PLAN

VALUES(4,4,1,2011);

INSERT INTO WORKOUT\_PLAN

VALUES(2,8,2,2003);

INSERT INTO WORKOUT\_PLAN

VALUES(5,7,3,2001);

INSERT INTO WORKOUT\_PLAN

VALUES(5,6,6,2012);

INSERT INTO WORKOUT\_PLAN

VALUES(7,9,5,2006);

INSERT INTO WORKOUT\_PLAN

VALUES(3,10,7,2009);

INSERT INTO WORKOUT\_PLAN

VALUES(6,5,10,2004);

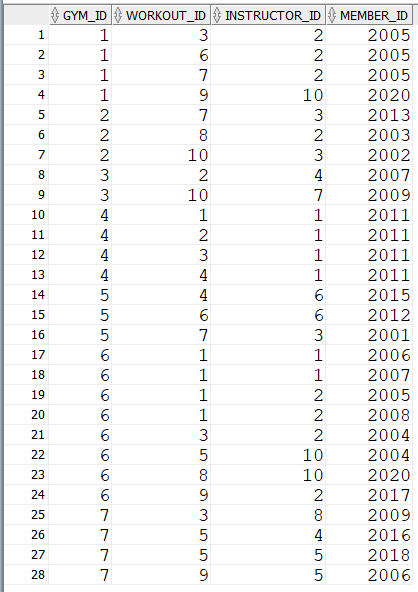
INSERT INTO WORKOUT\_PLAN

VALUES(6,8,10,2020);

INSERT INTO WORKOUT\_PLAN

VALUES(6,9,2,2017);

SELECT \* FROM WORKOUT\_PLAN;



--TABELA ASOCIATIVA BOOKS

INSERT INTO BOOKS

VALUES(2003,1001);

INSERT INTO BOOKS

VALUES(2010,1011);

INSERT INTO BOOKS

VALUES(2004,1002);

INSERT INTO BOOKS

VALUES(2019,1007);

INSERT INTO BOOKS

VALUES(2007,1006);

INSERT INTO BOOKS

VALUES(2020,1005);

INSERT INTO BOOKS

VALUES(2020,1009);

INSERT INTO BOOKS

VALUES(2018,1003);

INSERT INTO BOOKS

VALUES(2012,1010);

INSERT INTO BOOKS

VALUES(2011,1011);

INSERT INTO BOOKS

VALUES(2001,1004);

INSERT INTO BOOKS

VALUES(2008,1012);

INSERT INTO BOOKS

VALUES(2016,1004);

INSERT INTO BOOKS

VALUES(2017,1007);

INSERT INTO BOOKS

VALUES(2015,1012);

INSERT INTO BOOKS

VALUES(2010,1002);

INSERT INTO BOOKS

VALUES(2005,1010);

INSERT INTO BOOKS

VALUES(2010,1001);

INSERT INTO BOOKS

VALUES(2002,1007);

INSERT INTO BOOKS

VALUES(2012,1008);

INSERT INTO BOOKS

VALUES(2018,1003);

INSERT INTO BOOKS

VALUES(2020,1009);

INSERT INTO BOOKS

VALUES(2013,1011);

INSERT INTO BOOKS

VALUES(2019,1006);

INSERT INTO BOOKS

VALUES(2011,1003);

INSERT INTO BOOKS

VALUES(2018,1008);

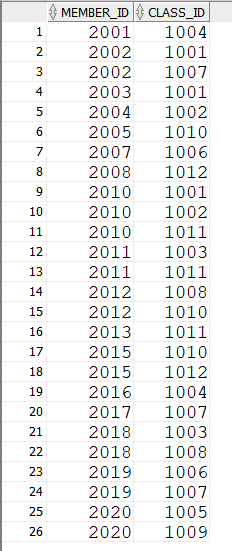
INSERT INTO BOOKS

VALUES(2015,1010);

INSERT INTO BOOKS

VALUES(2002,1001);

SELECT \* FROM BOOKS;



--TABELA PAYMENTS

INSERT INTO PAYMENTS

VALUES(9001,199,SYSDATE,2001);

INSERT INTO PAYMENTS

VALUES(9002,299,SYSDATE,2002);

INSERT INTO PAYMENTS

VALUES(9003,259,SYSDATE,2003);

INSERT INTO PAYMENTS

VALUES(9004,1990,SYSDATE,2004);

INSERT INTO PAYMENTS

VALUES(9005,2990,SYSDATE,2005);

INSERT INTO PAYMENTS

VALUES(9006,199,SYSDATE,2006);

INSERT INTO PAYMENTS

VALUES(9007,199,SYSDATE,2007);

INSERT INTO PAYMENTS

VALUES(9008,3199,SYSDATE,2008);

INSERT INTO PAYMENTS

VALUES(9009,299,SYSDATE,2009);

INSERT INTO PAYMENTS

VALUES(9010,259,SYSDATE,2010);

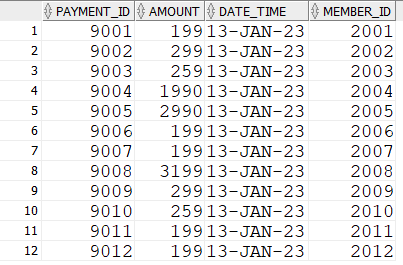
INSERT INTO PAYMENTS

VALUES(9011,199,SYSDATE,2011);

INSERT INTO PAYMENTS

VALUES(9012,199,SYSDATE,2012);

SELECT \* FROM PAYMENTS;



1. ***Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri diferite de colecții studiate. Apelați subprogramul.***

--Definiti un subprogram prin care sa obtineti nr de membrii antrenati de un instructor

--dat de la tasatura. Tratati toate exceptiile

--folosim functie stocata,un vector si un tablou imbricat

CREATE OR REPLACE type tablou\_imbricat is TABLE OF NUMBER(10);

/

CREATE OR REPLACE type vector1 AS VARRAY(10) OF NUMBER(10);

/

CREATE OR REPLACE FUNCTION f1(iid instructors.instructor\_id%TYPE)

RETURN number

IS t tablou\_imbricat := tablou\_imbricat();

v\_id vector1:=vector1();

x NUMBER(10);

rez number(10);

j number(10):=0;

BEGIN

FOR i in 1..5 LOOP

v\_id.extend;

SELECT instructor\_id into x

from INSTRUCTORS

where instructor\_id=i;

v\_id(i):=x;

END LOOP;

t.extend;

SELECT DISTINCT member\_id bulk collect into t

FROM WORKOUT\_PLAN

WHERE instructor\_id=iid;

rez:=t.count();

if rez=0 then

RAISE NO\_DATA\_FOUND;

ELSE

RETURN rez;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista membrii antrenati de instructorul dat');

END f1;

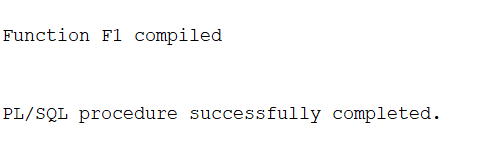
/

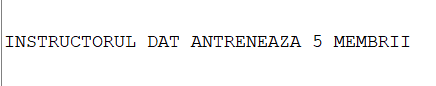
BEGIN

DBMS\_OUTPUT.PUT\_LINE('INSTRUCTORUL DAT ANTRENEAZA '|| f1('&p\_id') || ' MEMBRII');

END;

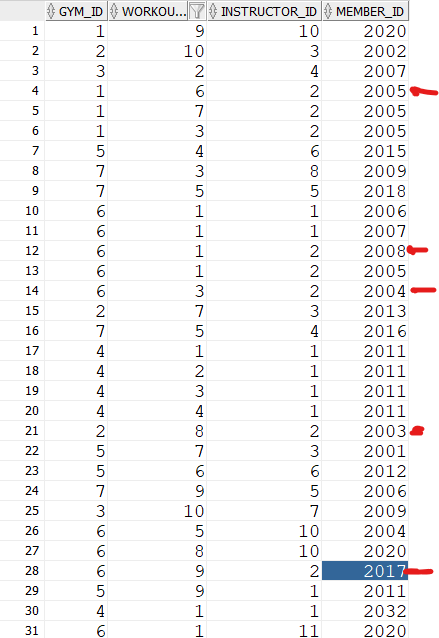
/





Input: 2

Output: 5



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

*--Pentru fiecare sala(id dat de la tastatura) sa se afiseze lista antrenamentelor ce au loc in acea sala*

*--folosim o procedura stocata, un ciclu cursor cu subcereri si un cursor parametrizat*

CREATE OR REPLACE PROCEDURE p1

is

CURSOR C (parametru gyms.gym\_id%TYPE)IS

SELECT WORKOUT\_NAME,INSTRUCTOR\_NAME,MEMBER\_NAME

FROM WORKOUT W, INSTRUCTORS p, MEMBERS M, WORKOUT\_PLAN WP

WHERE WP.GYM\_ID=parametru AND W.WORKOUT\_ID=WP.WORKOUT\_ID AND P.INSTRUCTOR\_ID=WP.INSTRUCTOR\_ID AND M.MEMBER\_ID=WP.MEMBER\_ID;

BEGIN

FOR I IN (SELECT GYM\_ID FROM GYMS)LOOP

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

DBMS\_OUTPUT.PUT\_LINE ('GYM '||i.GYM\_ID);

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

FOR J IN C(i.gym\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE ('MEMBRUL '|| j.MEMBER\_NAME ||' ARE WORKOUT-UL ' || j.WORKOUT\_NAME ||' CU INSTRUCTORUL ' || j.INSTRUCTOR\_NAME);

END LOOP;

END LOOP;

END p1;

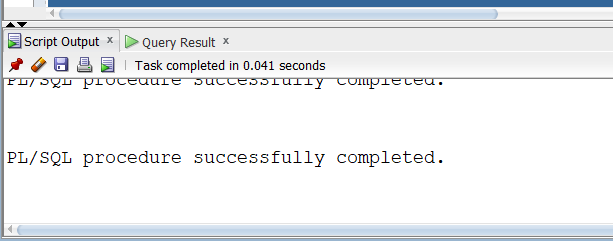
/

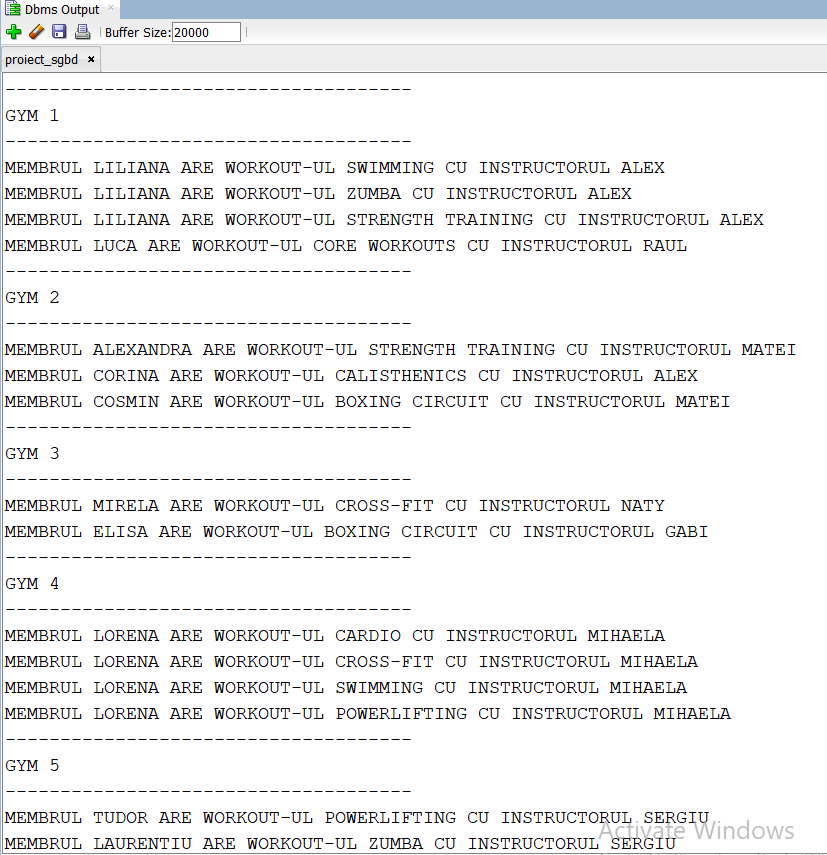
BEGIN

p1;

END;

/





1. ***Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.***

*--sa se afiseze numarul de angajati dintr-o sala data de la tastatura*

*--folosim o functie stocata*

CREATE OR REPLACE FUNCTION F2(v\_id gyms.gym\_id%TYPE)

RETURN NUMBER

is A NUMBER;

B NUMBER;

C NUMBER;

rez NUMBER;

begin

SELECT COUNT(distinct G.EMPLOYEE\_ID), COUNT(distinct c.EMPLOYEE\_ID),COUNT(distinct r.EMPLOYEE\_ID)

into A,B,C

FROM GUARDIANS G, CLEANING\_STAFF C, RECEPTIONISTS R

WHERE G.GYM\_ID=C.GYM\_ID AND G.GYM\_ID=R.GYM\_ID AND G.GYM\_ID=V\_ID

group by g.gym\_id

ORDER BY G.GYM\_ID ASC;

rez:=a+b+c;

return rez;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista aceasta sala');

WHEN TOO\_MANY\_ROWS THEN RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multe sali cu acest id');

WHEN OTHERS THEN RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

end f2;

/

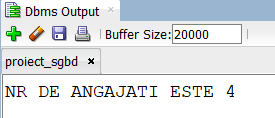
DROP FUNCTION F2;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('NR DE ANGAJATI ESTE ' || f2('&P'));

END;





--8 refacut

--sa se calculeze numarul de angajati ai unei sali cu dimensiunea data de la tastatura

--si sa se stearga aceasta sala

CREATE OR REPLACE FUNCTION f2\_pmm(dim gyms.dimension%TYPE)

RETURN VARCHAR IS MSG VARCHAR(40);

A NUMBER;

B NUMBER;

C NUMBER;

txt VARCHAR(40):='SALA A FOST STEARSA CU SUCCES';

copie gyms.dimension%TYPE;

NR\_ANG NUMBER;

nd NUMBER;

exista\_ang EXCEPTION;

PRAGMA EXCEPTION\_INIT(exista\_ang,-2292); --EXCEPTIE INTERNA NEPREDEFINITA

exc1 EXCEPTION;

exc2 EXCEPTION;

BEGIN

SELECT COUNT(DIMENSION) INTO nd

from gyms

where dimension=dim;

if nd=0 then

RAISE EXC1;

ELSIF nd>1 then

RAISE EXC2;

end if;

BEGIN

SELECT COUNT(distinct G.EMPLOYEE\_ID), COUNT(distinct c.EMPLOYEE\_ID),COUNT(distinct r.EMPLOYEE\_ID)

into A,B,C

FROM GUARDIANS G, CLEANING\_STAFF C, RECEPTIONISTS R, GYMS GM

WHERE G.GYM\_ID=C.GYM\_ID AND G.GYM\_ID=R.GYM\_ID AND GM.GYM\_ID=G.GYM\_ID AND GM.DIMENSION=DIM

group by g.gym\_id

ORDER BY G.GYM\_ID ASC;

NR\_ANG:=a+b+c;

MSG:=CONCAT('NUMARUL DE ANGAJATI ESTE ', TO\_CHAR(NR\_ANG));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('NU EXISTA ANGAJATI LA ACEASTA SALA');

END;

DELETE FROM GYMS WHERE DIMENSION=dim;

RETURN txt;

EXCEPTION

WHEN EXC1 THEN DBMS\_OUTPUT.PUT\_LINE('NU AVEM O SALA CU DIMENSIUNEA DATA');

RETURN MSG;

WHEN EXC2 THEN DBMS\_OUTPUT.PUT\_LINE('EXISTA MAI MULTE SALI CU DIMENSIUNEA DATA');

RETURN MSG;

WHEN exista\_ang THEN DBMS\_OUTPUT.PUT\_LINE('NU PUTETI STERGE ABONAMENTUL DEOARECE EXISTA MEMBRII CE AU ACEASTA SUBSCRIPTIE');

RETURN MSG;

END F2\_PMM;

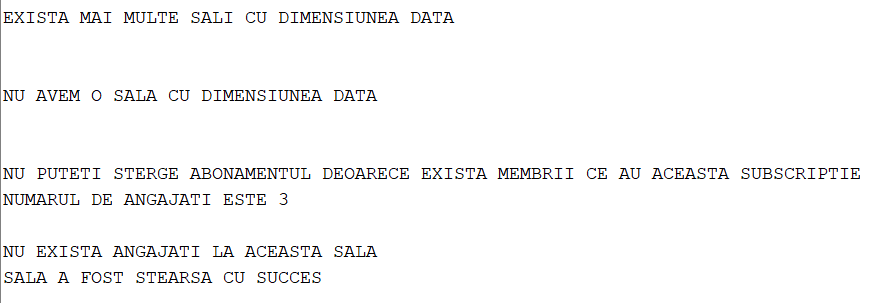
/

--TOO MANY ROWS: 10

--NO DATA FOUND: 35

--FOREIGN KEY RESTRICTION : 80

--SUCCES:70



1. ***Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.***

*--sa se afiseze numarul de antrenamente pe care il are un membru dat de la tastatura(NUMELE SAU)*

*--cu un anumit instructor dat de la tastatura(numele sau)*

*--la o sala data de la tastatura*

*-- si cu abonament de tip SILVER 1 luna*

*--folosim procedura stocata SI CURSOR EXPLICIT*

CREATE OR REPLACE PROCEDURE P2(m\_nume members.member\_name%TYPE,

i\_nume instructors.instructor\_name%TYPE,

g\_id gyms.gym\_id%TYPE,

ms\_nume membership.membership\_name%TYPE,

ms\_id membership.membership\_id%TYPE)

is

rez NUMBER;

a\_nume members.member\_name%TYPE;

b\_nume instructors.instructor\_name%TYPE;

c\_id gyms.gym\_id%TYPE;

CURSOR C IS

SELECT COUNT(\*)

FROM WORKOUT\_PLAN WK,MEMBERS M ,INSTRUCTORS I, GYMS G , MEMBERSHIP MS

WHERE upper(M.MEMBER\_NAME)=upper(m\_nume) AND WK.MEMBER\_ID=M.MEMBER\_ID and

upper(I.instructor\_name)=upper(i\_nume) AND WK.INSTRUCTOR\_ID=I.INSTRUCTOR\_ID AND

WK.GYM\_ID=g\_id AND WK.GYM\_ID=G.GYM\_ID AND

upper(MS.MEMBERSHIP\_NAME)=upper(ms\_nume) AND MS.MEMBERSHIP\_ID=ms\_id;

BEGIN

select member\_name

into a\_nume

from members

where member\_name=m\_nume;

OPEN C;

LOOP

FETCH C INTO REZ;

EXIT WHEN C%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('NR DE ANTRENAMENTE FACUTE DE ' || M\_NUME ||' CU ' || I\_NUME || ' IN SALA ' || G\_ID|| ' ESTE ' || REZ);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista acest membru/instructor/sala/abonament');

WHEN TOO\_MANY\_ROWS THEN RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multi membri/instructori/sali/memberships cu acest nume');

WHEN OTHERS THEN RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

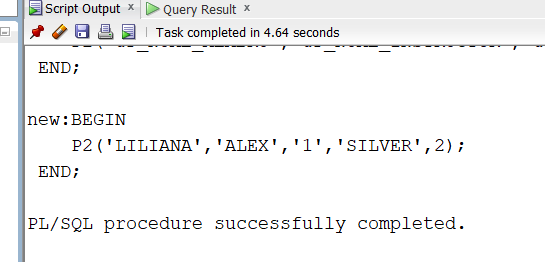
END P2;

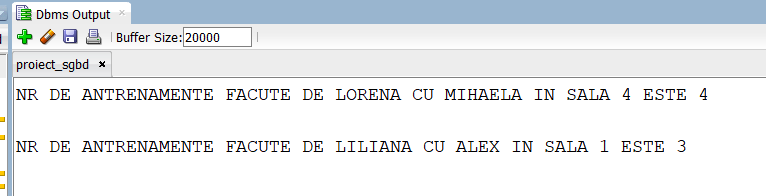
/

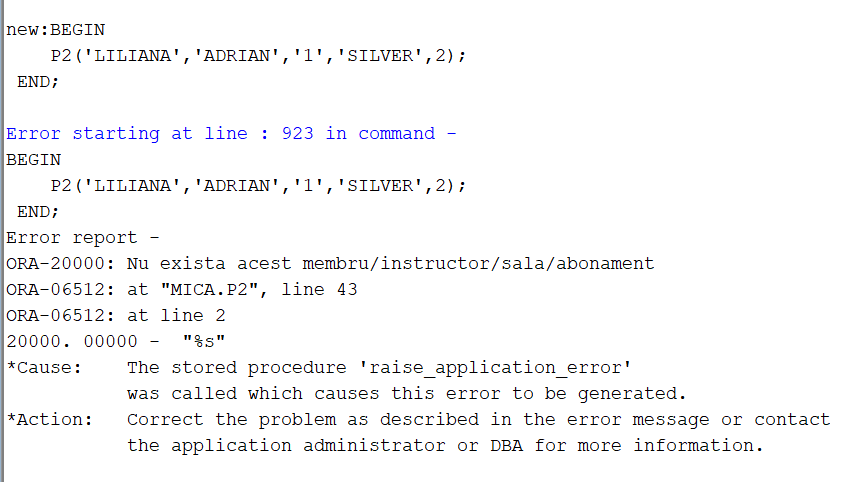
BEGIN

P2('&P\_NUME\_MEMBRU','&P\_NUME\_INSTRUCTOR','&P\_ID\_SALA','SILVER',2);

END;







--9 refacut

--EX9

--sa se afiseze numarul de antrenamente pe care il are un membru dat de la tastatura(NUMELE SAU)

--cu un anumit instructor dat de la tastatura(numele sau)

--la o sala data de la tastatura

-- si cu abonament de tip SILVER 1 luna (membership cod 2)

--folosim procedura stocata SI CURSOR EXPLICIT

CREATE OR REPLACE PROCEDURE P2 (m\_nume members.member\_name%TYPE,

i\_nume instructors.instructor\_name%TYPE,

g\_id gyms.gym\_id%TYPE,

ms\_id membership.membership\_id%TYPE)

is

rez NUMBER;

a NUMBER;

b NUMBER;

c\_id NUMBER;

d number;

exc1 exception;

exc2 exception;

exc3 exception;

exc4 exception;

CURSOR C IS

SELECT COUNT(\*)

FROM WORKOUT\_PLAN WK,MEMBERS M ,INSTRUCTORS I, GYMS G , MEMBERSHIP MS

WHERE

upper(M.MEMBER\_NAME)=upper(m\_nume) AND

WK.MEMBER\_ID=M.MEMBER\_ID and

upper(I.instructor\_name)=upper(i\_nume) AND

WK.INSTRUCTOR\_ID=I.INSTRUCTOR\_ID AND

WK.GYM\_ID=g\_id AND WK.GYM\_ID=G.GYM\_ID AND

MS.MEMBERSHIP\_ID=ms\_id;

BEGIN

select count(member\_id )

into a

from members

where UPPER(member\_name)=UPPER(m\_nume);

if a = 0 then

RAISE exc1;

elsif a>1 then

RAISE TOO\_MANY\_ROWS;

END IF;

select count(INSTRUCTOR\_ID )

into b

from INSTRUCTORS

where UPPER(INSTRUCTOR\_NAME)=UPPER(I\_NUME);

if b = 0 then

RAISE exc2;

ELSIF B>1 THEN

RAISE TOO\_MANY\_ROWS;

END IF;

select count(gym\_id)

into c\_id

from gyms

where gym\_id=g\_id;

if c\_id = 0 then

RAISE exc3;

END IF;

select membership\_id

into d

from membership

where membership\_id=ms\_id;

if d != 2 then

RAISE EXC4;

END IF;

OPEN C;

LOOP

FETCH C INTO REZ;

if REZ = 0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXIT WHEN C%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('NR DE ANTRENAMENTE FACUTE DE ' || M\_NUME ||' CU ' || I\_NUME || ' IN SALA ' || G\_ID|| ' ESTE ' || REZ);

END LOOP;

EXCEPTION

WHEN exc1 THEN RAISE\_APPLICATION\_ERROR(-20003,'Nu exista acest membru');

WHEN exc2 THEN RAISE\_APPLICATION\_ERROR(-20010,'Nu exista acest instructor');

WHEN exc3 THEN RAISE\_APPLICATION\_ERROR(-20100,'Nu exista aceasta sala');

WHEN exc4 THEN RAISE\_APPLICATION\_ERROR(-20104,'Abonamentul nu este de tip SILVER 1 MONTH');

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista acest plan de antrenament');

WHEN TOO\_MANY\_ROWS THEN RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multi membri/instructori cu acest nume');

END P2;

/

BEGIN

P2('&P\_NUME\_MEMBRU','&P\_NUME\_INSTRUCTOR','&P\_ID\_SALA','&p\_id\_membership');

END;

-- NO DATA FOUND MEMBRU : CORNEL, MIHAELA , 2 ,2

--TOO MANY ROWS MEMBRU : LORENA, MIHAELA, 4, 2

--NO DATA FOUND INSTRUCTOR: ADRIAN, CORNEL, 3, 2

--TOO MANY ROWS INSTRUCTOR: LUCA, RAUL, 2 2

--ABONAMENT GRESIT : LILIANA MIHAELA 4 1

--NO DATA FOUND PLAN: LILIANA ALEX 5 2

--SUCCES : LILIANA ALEX 1 2

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

--Definiti un declansator care sa nu permita inserarea unui gardian care nu exista in tabela employees sau care exista deja in tabela cleaning\_staff sau receptionists.

--Definiti un declansator care sa nu permita modificarea tabelei workout\_plan de altcineva inafara de instructor

*--Definiti un declansator care sa nu permita modificarea tabelei workout\_plan in afara intervalului 10:00-16:00*

CREATE OR REPLACE TRIGGER trig\_lmd1

BEFORE INSERT OR DELETE OR UPDATE on WORKOUT\_PLAN

BEGIN

IF (TO\_CHAR(SYSDATE,'D') = 1) OR (TO\_CHAR(SYSDATE,'HH24') NOT BETWEEN 10 AND 16)THEN

IF INSERTING THEN RAISE\_APPLICATION\_ERROR(-20001,'Inserarea in WORKOUT\_PLAN este permisa doar in timpul programului de lucru!');

ELSIF DELETING THEN RAISE\_APPLICATION\_ERROR(-20002,'Stergerea din WORKOUT\_PLAN este permisa doar in timpul programului de lucru!');

ELSE RAISE\_APPLICATION\_ERROR(-20003,'Actualizarile in WORKOUT\_PLAN sunt permise doar in timpul programului de lucru!');

END IF;

END IF;

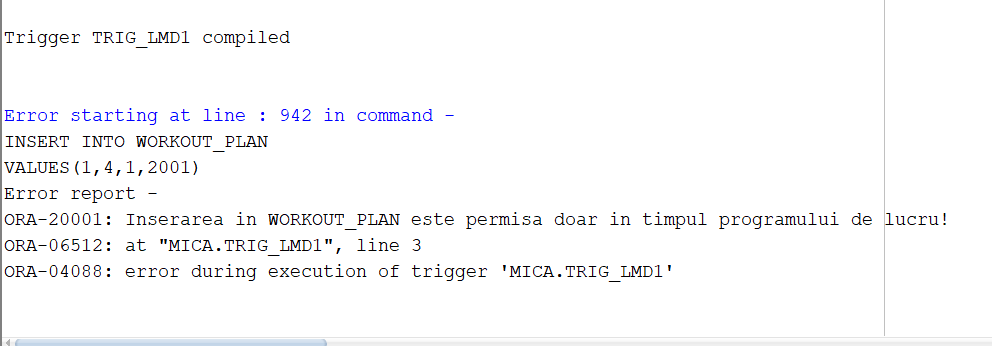
END;

/

DROP TRIGGER trig\_lmd1;

INSERT INTO WORKOUT\_PLAN

VALUES(1,4,1,2001);



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

*--Definiti un declansator care sa nu permita inserarea unui membru a carui varsta este mai mica sau egala cu 15 ani si a carui data de incheiere a abonamentului este inaintea inregistrarii sale*

CREATE OR REPLACE TRIGGER trig\_lmd2

BEFORE INSERT ON MEMBERS

FOR EACH ROW

BEGIN

if :NEW.AGE<=15 THEN

RAISE\_APPLICATION\_ERROR(-20910, 'Eroare: Varsta trebuie sa fie peste 15 ani!');

elsif :NEW.ending\_date<:NEW.joining\_date THEN

RAISE\_APPLICATION\_ERROR(-20900, 'Eroare: ENDING\_DATE nu poate fi inaintea JOINING\_DATE!');

Else

DBMS\_OUTPUT.PUT\_LINE('INSERARE REUSITA!');

END IF;

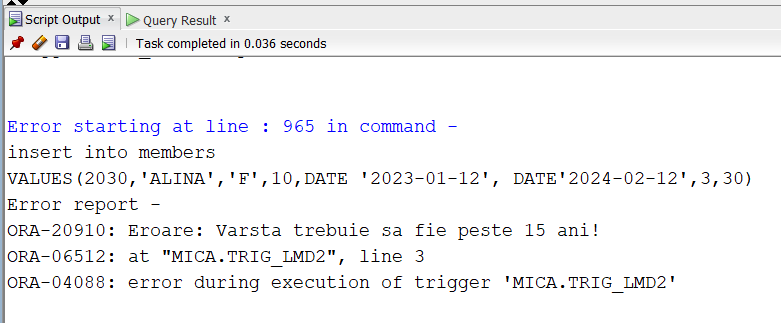
END;

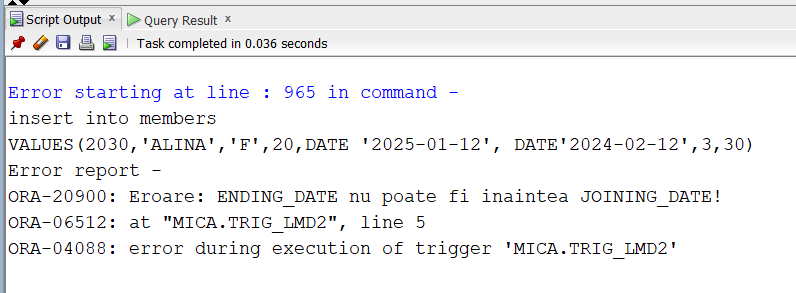
/

insert into members

VALUES(2030,'ALINA','F',20,DATE '2025-01-12', DATE'2024-02-12',3,30);

DELETE FROM MEMBERS WHERE MEMBER\_NAME='ALINA';





1. *Definiți un trigger de tip LDD. Declanșați trigger-ul.*

*----Definiti un declansator LDD care sa nu permita crearea, stergerea sau modificarea tabelelor pana la dezactivarea acestuia*

CREATE OR REPLACE TRIGGER trig\_ldd

BEFORE CREATE OR ALTER OR DROP ON DATABASE

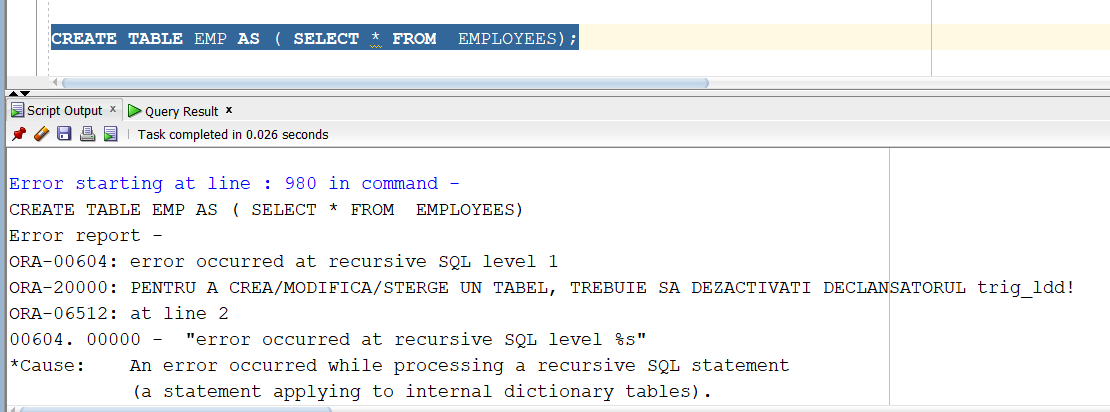
BEGIN

RAISE\_APPLICATION\_ERROR(-20000, 'PENTRU A CREA/MODIFICA/STERGE UN TABEL, TREBUIE SA DEZACTIVATI DECLANSATORUL trig\_ldd!');

END;

/

CREATE TABLE EMP AS ( SELECT \* FROM EMPLOYEES);

**

1. ***Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.***

CREATE OR REPLACE PACKAGE pachet1 AS

FUNCTION F1

RETURN NUMBER;

FUNCTION F2(v\_id gyms.gym\_id%TYPE)

RETURN NUMBER;

PROCEDURE P1;

PROCEDURE P2(m\_nume members.member\_name%TYPE,

i\_nume instructors.instructor\_name%TYPE,

g\_id gyms.gym\_id%TYPE,

ms\_nume membership.membership\_name%TYPE,

ms\_id membership.membership\_id%TYPE);

END pachet1;

/

CREATE OR REPLACE PACKAGE BODY pachet1 AS

FUNCTION F1

RETURN number

IS t tablou\_imbricat := tablou\_imbricat();

v\_id vector1:=vector1();

x NUMBER(10);

rez number(10);

j number(10):=0;

BEGIN

FOR i in 1..5 LOOP

v\_id.extend;

SELECT instructor\_id into x

from INSTRUCTORS

where instructor\_id=i;

v\_id(i):=x;

END LOOP;

j:=v\_id(3);

t.extend;

SELECT member\_id bulk collect into t

FROM WORKOUT\_PLAN

WHERE instructor\_id=j;

rez:=t.count();

RETURN rez;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista membrii antrenati de instructorul dat');

WHEN TOO\_MANY\_ROWS THEN RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multi membrii antrenati de mai multi instructori!');

WHEN OTHERS THEN RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

End;

FUNCTION F2(v\_id gyms.gym\_id%TYPE)

RETURN NUMBER

is A NUMBER;

B NUMBER;

C NUMBER;

rez NUMBER;

begin

SELECT COUNT(distinct G.EMPLOYEE\_ID), COUNT(distinct c.EMPLOYEE\_ID),COUNT(distinct r.EMPLOYEE\_ID)

into A,B,C

FROM GUARDIANS G, CLEANING\_STAFF C, RECEPTIONISTS R

WHERE G.GYM\_ID=C.GYM\_ID AND G.GYM\_ID=R.GYM\_ID AND G.GYM\_ID=V\_ID

group by g.gym\_id

ORDER BY G.GYM\_ID ASC;

rez:=a+b+c;

return rez;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista aceasta sala');

WHEN TOO\_MANY\_ROWS THEN RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multe sali cu acest id');

WHEN OTHERS THEN RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

end;

PROCEDURE P1

is

CURSOR C (parametru gyms.gym\_id%TYPE)IS

SELECT WORKOUT\_NAME,INSTRUCTOR\_NAME,MEMBER\_NAME

FROM WORKOUT W, INSTRUCTORS p, MEMBERS M, WORKOUT\_PLAN WP

WHERE WP.GYM\_ID=parametru AND W.WORKOUT\_ID=WP.WORKOUT\_ID AND P.INSTRUCTOR\_ID=WP.INSTRUCTOR\_ID AND M.MEMBER\_ID=WP.MEMBER\_ID;

BEGIN

FOR I IN (SELECT GYM\_ID FROM GYMS)LOOP

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

DBMS\_OUTPUT.PUT\_LINE ('GYM '||i.GYM\_ID);

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

FOR J IN C(i.gym\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE ('MEMBRUL '|| j.MEMBER\_NAME ||' ARE WORKOUT-UL ' || j.WORKOUT\_NAME ||' CU INSTRUCTORUL ' || j.INSTRUCTOR\_NAME);

END LOOP;

END LOOP;

END ;

PROCEDURE P2(m\_nume members.member\_name%TYPE,

i\_nume instructors.instructor\_name%TYPE,

g\_id gyms.gym\_id%TYPE,

ms\_nume membership.membership\_name%TYPE,

ms\_id membership.membership\_id%TYPE)

is

rez NUMBER;

a\_nume members.member\_name%TYPE;

b\_nume instructors.instructor\_name%TYPE;

c\_id gyms.gym\_id%TYPE;

CURSOR C IS

SELECT COUNT(\*)

FROM WORKOUT\_PLAN WK,MEMBERS M ,INSTRUCTORS I, GYMS G , MEMBERSHIP MS

WHERE upper(M.MEMBER\_NAME)=upper(m\_nume) AND WK.MEMBER\_ID=M.MEMBER\_ID and

upper(I.instructor\_name)=upper(i\_nume) AND WK.INSTRUCTOR\_ID=I.INSTRUCTOR\_ID AND

WK.GYM\_ID=g\_id AND WK.GYM\_ID=G.GYM\_ID AND

upper(MS.MEMBERSHIP\_NAME)=upper(ms\_nume) AND MS.MEMBERSHIP\_ID=ms\_id;

BEGIN

select member\_name

into a\_nume

from members

where member\_name=m\_nume;

OPEN C;

LOOP

FETCH C INTO REZ;

EXIT WHEN C%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('NR DE ANTRENAMENTE FACUTE DE ' || M\_NUME ||' CU ' || I\_NUME || ' IN SALA ' || G\_ID|| ' ESTE ' || REZ);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista acest membru/instructor/sala/abonament');

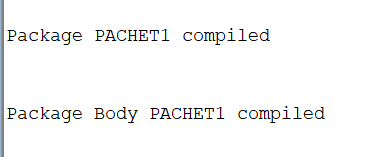
WHEN TOO\_MANY\_ROWS THEN RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multi membri/instructori/sali/memberships cu acest nume');

WHEN OTHERS THEN RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

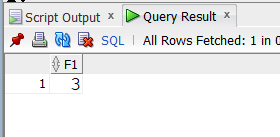
END ;

end pachet1;

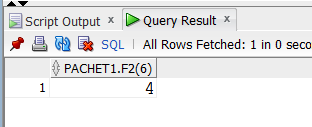
/



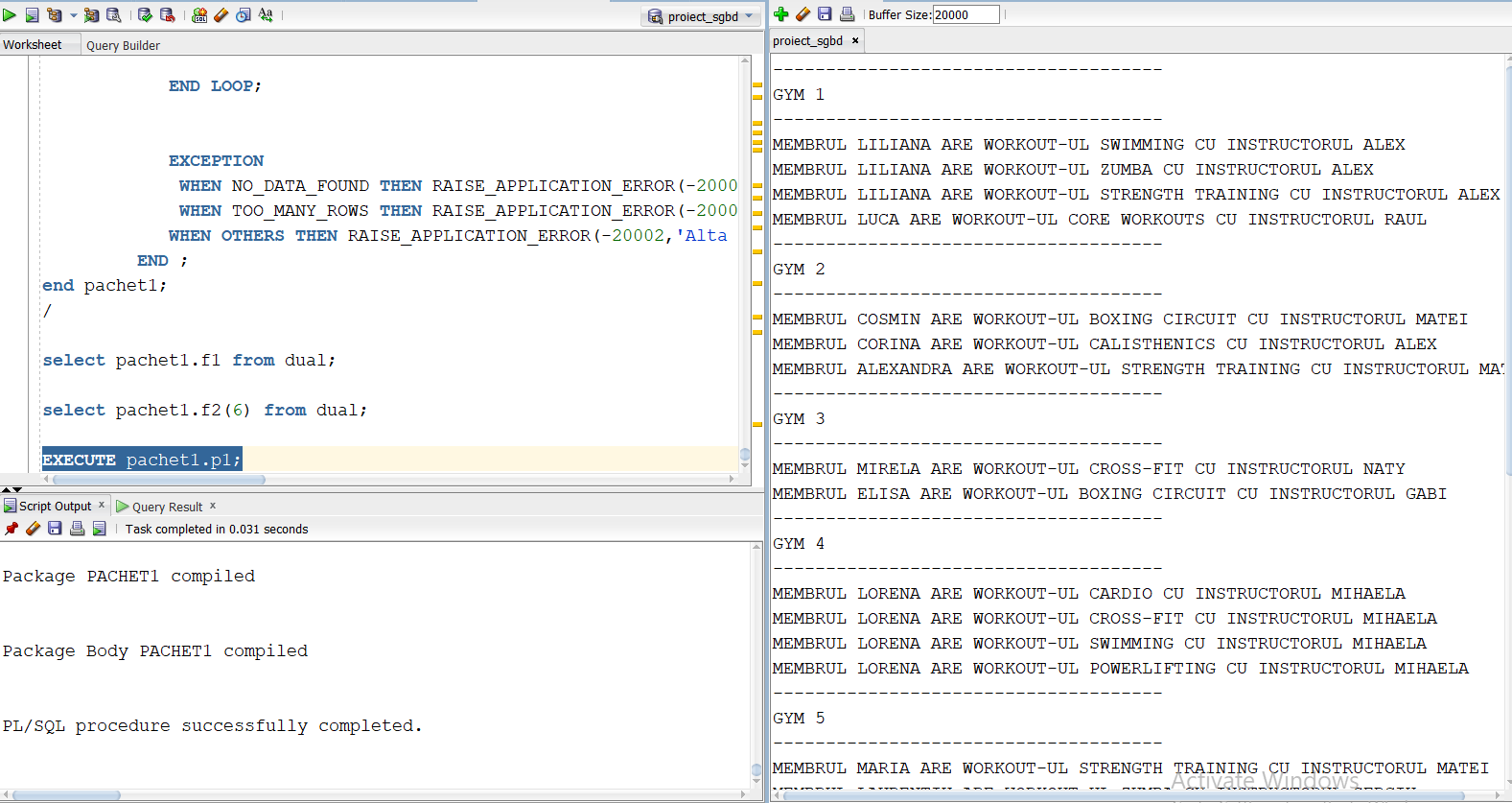
select pachet1.f1 from dual;



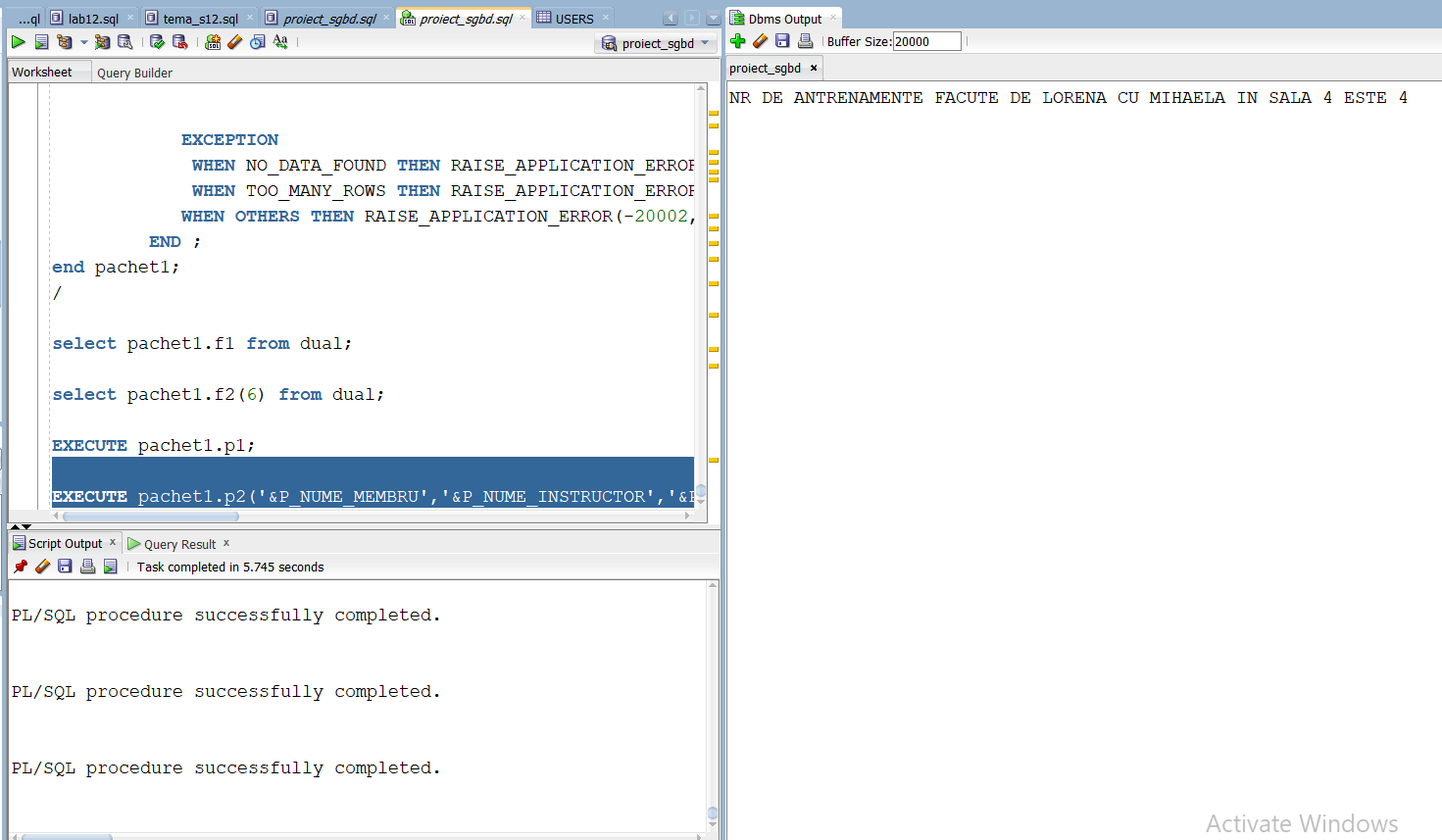
select pachet1.f2(6) from dual;



EXECUTE pachet1.p1;



EXECUTE pachet1.p2('&P\_NUME\_MEMBRU','&P\_NUME\_INSTRUCTOR','&P\_ID\_SALA','SILVER',2);



--13 refacut

CREATE OR REPLACE PACKAGE ex13 AS

FUNCTION F1(iid instructors.instructor\_id%TYPE)

RETURN NUMBER;

FUNCTION f2\_pmm(dim gyms.dimension%TYPE)

RETURN VARCHAR;

PROCEDURE P1;

PROCEDURE P2 (m\_nume members.member\_name%TYPE,

i\_nume instructors.instructor\_name%TYPE,

g\_id gyms.gym\_id%TYPE,

ms\_id membership.membership\_id%TYPE);

END ex13;

/

CREATE OR REPLACE PACKAGE BODY EX13 AS

FUNCTION F1(iid instructors.instructor\_id%TYPE)

RETURN number

IS t tablou\_imbricat := tablou\_imbricat();

v\_id vector1:=vector1();

x NUMBER(10);

rez number(10);

j number(10):=0;

BEGIN

FOR i in 1..5 LOOP

v\_id.extend;

SELECT instructor\_id into x

from INSTRUCTORS

where instructor\_id=i;

v\_id(i):=x;

END LOOP;

t.extend;

SELECT DISTINCT member\_id bulk collect into t

FROM WORKOUT\_PLAN

WHERE instructor\_id=iid;

rez:=t.count();

if rez=0 then

RAISE NO\_DATA\_FOUND;

ELSE

RETURN rez;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista membrii antrenati de instructorul dat');

END f1;

FUNCTION f2\_pmm(dim gyms.dimension%TYPE)

RETURN VARCHAR IS MSG VARCHAR(40);

A NUMBER;

B NUMBER;

C NUMBER;

txt VARCHAR(40):='SALA A FOST STEARSA CU SUCCES';

copie gyms.dimension%TYPE;

NR\_ANG NUMBER;

nd NUMBER;

exista\_ang EXCEPTION;

PRAGMA EXCEPTION\_INIT(exista\_ang,-2292);

exc1 EXCEPTION;

exc2 EXCEPTION;

BEGIN

SELECT COUNT(DIMENSION) INTO nd

from gyms

where dimension=dim;

if nd=0 then

RAISE EXC1;

ELSIF nd>1 then

RAISE EXC2;

end if;

BEGIN

SELECT COUNT(distinct G.EMPLOYEE\_ID), COUNT(distinct c.EMPLOYEE\_ID),COUNT(distinct r.EMPLOYEE\_ID)

into A,B,C

FROM GUARDIANS G, CLEANING\_STAFF C, RECEPTIONISTS R, GYMS GM

WHERE G.GYM\_ID=C.GYM\_ID AND G.GYM\_ID=R.GYM\_ID AND GM.GYM\_ID=G.GYM\_ID AND GM.DIMENSION=DIM

group by g.gym\_id

ORDER BY G.GYM\_ID ASC;

NR\_ANG:=a+b+c;

MSG:=CONCAT('NUMARUL DE ANGAJATI ESTE ', TO\_CHAR(NR\_ANG));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('NU EXISTA ANGAJATI LA ACEASTA SALA');

END;

--DELETE FROM GYMS WHERE DIMENSION=dim;

RETURN msg;

EXCEPTION

WHEN EXC1 THEN DBMS\_OUTPUT.PUT\_LINE('NU AVEM O SALA CU DIMENSIUNEA DATA');

msg:='NU AVEM O SALA CU DIMENSIUNEA DATA';

RETURN MSG;

WHEN EXC2 THEN DBMS\_OUTPUT.PUT\_LINE('EXISTA MAI MULTE SALI CU DIMENSIUNEA DATA');

msg:='EXISTA MAI MULTE SALI CU DIMENSIUNEA DATA';

RETURN MSG;

WHEN exista\_ang THEN DBMS\_OUTPUT.PUT\_LINE('NU PUTETI STERGE ABONAMENTUL DEOARECE EXISTA MEMBRII CE AU ACEASTA SUBSCRIPTIE');

msg:='NU PUTETI STERGE ABONAMENTUL DEOARECE EXISTA MEMBRII CE AU ACEASTA SUBSCRIPTIE';

RETURN MSG;

END F2\_PMM;

PROCEDURE P1

is

CURSOR C (parametru gyms.gym\_id%TYPE)IS

SELECT WORKOUT\_NAME,INSTRUCTOR\_NAME,MEMBER\_NAME

FROM WORKOUT W, INSTRUCTORS p, MEMBERS M, WORKOUT\_PLAN WP

WHERE WP.GYM\_ID=parametru AND W.WORKOUT\_ID=WP.WORKOUT\_ID AND P.INSTRUCTOR\_ID=WP.INSTRUCTOR\_ID AND M.MEMBER\_ID=WP.MEMBER\_ID;

BEGIN

FOR I IN (SELECT GYM\_ID FROM GYMS)LOOP

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

DBMS\_OUTPUT.PUT\_LINE ('GYM '||i.GYM\_ID);

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

FOR J IN C(i.gym\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE ('MEMBRUL '|| j.MEMBER\_NAME ||' ARE WORKOUT-UL ' || j.WORKOUT\_NAME ||' CU INSTRUCTORUL ' || j.INSTRUCTOR\_NAME);

END LOOP;

END LOOP;

END ;

PROCEDURE P2 (m\_nume members.member\_name%TYPE,

i\_nume instructors.instructor\_name%TYPE,

g\_id gyms.gym\_id%TYPE,

ms\_id membership.membership\_id%TYPE)

is

rez NUMBER;

a NUMBER;

b NUMBER;

c\_id NUMBER;

d number;

exc1 exception;

exc2 exception;

exc3 exception;

exc4 exception;

CURSOR C IS

SELECT COUNT(\*)

FROM WORKOUT\_PLAN WK,MEMBERS M ,INSTRUCTORS I, GYMS G , MEMBERSHIP MS

WHERE

upper(M.MEMBER\_NAME)=upper(m\_nume) AND

WK.MEMBER\_ID=M.MEMBER\_ID and

upper(I.instructor\_name)=upper(i\_nume) AND

WK.INSTRUCTOR\_ID=I.INSTRUCTOR\_ID AND

WK.GYM\_ID=g\_id AND WK.GYM\_ID=G.GYM\_ID AND

MS.MEMBERSHIP\_ID=ms\_id;

BEGIN

select count(member\_id )

into a

from members

where UPPER(member\_name)=UPPER(m\_nume);

if a = 0 then

RAISE exc1;

elsif a>1 then

RAISE TOO\_MANY\_ROWS;

END IF;

select count(INSTRUCTOR\_ID )

into b

from INSTRUCTORS

where UPPER(INSTRUCTOR\_NAME)=UPPER(I\_NUME);

if b = 0 then

RAISE exc2;

ELSIF B>1 THEN

RAISE TOO\_MANY\_ROWS;

END IF;

select count(gym\_id)

into c\_id

from gyms

where gym\_id=g\_id;

if c\_id = 0 then

RAISE exc3;

END IF;

select membership\_id

into d

from membership

where membership\_id=ms\_id;

if d != 2 then

RAISE EXC4;

END IF;

OPEN C;

LOOP

FETCH C INTO REZ;

if REZ = 0 THEN

RAISE NO\_DATA\_FOUND;

END IF;

EXIT WHEN C%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('NR DE ANTRENAMENTE FACUTE DE ' || M\_NUME ||' CU ' || I\_NUME || ' IN SALA ' || G\_ID|| ' ESTE ' || REZ);

END LOOP;

EXCEPTION

WHEN exc1 THEN RAISE\_APPLICATION\_ERROR(-20003,'Nu exista acest membru');

WHEN exc2 THEN RAISE\_APPLICATION\_ERROR(-20010,'Nu exista acest instructor');

WHEN exc3 THEN RAISE\_APPLICATION\_ERROR(-20100,'Nu exista aceasta sala');

WHEN exc4 THEN RAISE\_APPLICATION\_ERROR(-20104,'Abonamentul nu este de tip SILVER 1 MONTH');

WHEN NO\_DATA\_FOUND THEN RAISE\_APPLICATION\_ERROR(-20000,'Nu exista acest plan de antrenament');

WHEN TOO\_MANY\_ROWS THEN RAISE\_APPLICATION\_ERROR(-20001,'Exista mai multi membri/instructori cu acest nume');

END P2;

end EX13;

/

select ex13.f1('&p\_id') from dual;

begin

dbms\_output.put\_line(ex13.f2\_pmm('&p\_dim'));

end;

EXECUTE ex13.p1;

EXECUTE ex13.p2('&P\_NUME\_MEMBRU','&P\_NUME\_INSTRUCTOR','&P\_ID\_SALA','&p\_id\_membership');

drop package ex13;

1. ***Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).***

CREATE OR REPLACE PACKAGE pachet2 is

type user\_details is record(

u\_id users.user\_id%TYPE,

f\_name users.first\_name%TYPE,

m\_id members.member\_id%TYPE);

type workouts is record(

m\_name members.member\_name%TYPE,

w\_name workout.workout\_name%TYPE,

i\_name instructors.instructor\_name%TYPE,

g\_id NUMBER(10));

FUNCTION get\_member\_name (mid members.member\_id%TYPE)

return VARCHAR;

FUNCTION get\_instructor\_name (iid instructors.instructor\_name%TYPE)

return VARCHAR;

FUNCTION get\_workout\_name (wid workout.workout\_name%TYPE)

return VARCHAR;

PROCEDURE add\_workout\_plan(gid gyms.gym\_id%TYPE,

wid workout.workout\_id%TYPE,

iid instructors.instructor\_id%TYPE,

mid members.member\_id%TYPE);

PROCEDURE modify\_payment(pid payments.payment\_id%TYPE,

am payments.amount%TYPE,

m\_id members.member\_id%TYPE,

new\_am payments.amount%TYPE

);

END pachet2;

/

DROP PACKAGE pachet2;

CREATE OR REPLACE PACKAGE BODY pachet2 IS

vu user\_details;

vw workouts;

FUNCTION get\_member\_name (mid members.member\_id%TYPE)

return VARCHAR is nume varchar(20);

BEGIN

SELECT MEMBER\_NAME INTO nume

FROM MEMBERS

WHERE MEMBER\_ID=mid;

return nume;

end;

FUNCTION get\_instructor\_name (iid instructors.instructor\_name%TYPE)

return VARCHAR is nume varchar(20);

BEGIN

SELECT instructor\_name INTO nume

FROM INSTRUCTORS

WHERE INSTRUCTOR\_ID=iid;

return nume;

end;

FUNCTION get\_workout\_name (wid workout.workout\_name%TYPE)

return VARCHAR is nume varchar(20);

BEGIN

SELECT workout\_name INTO nume

FROM WORKOUT

WHERE WORKOUT\_ID=wid;

return nume;

end;

PROCEDURE add\_workout\_plan(gid gyms.gym\_id%TYPE,

wid workout.workout\_id%TYPE,

iid instructors.instructor\_id%TYPE,

mid members.member\_id%TYPE) is

BEGIN

vw.m\_name:=get\_member\_name(2001);

vw.w\_name:=get\_workout\_name(4);

vw.i\_name:=get\_instructor\_name(10);

vw.g\_id:=6;

DBMS\_OUTPUT.PUT\_LINE('PLANUL DE ANTRENAMENT A FOST ADAUGAT');

DBMS\_OUTPUT.PUT\_LINE(vw.m\_name || ' FACE ' || vw.w\_name ||' CU INSTRUCTORUL ' || vw.i\_name|| ' IN SALA '|| vw.g\_id);

END add\_workout\_plan;

PROCEDURE modify\_payment(pid payments.payment\_id%TYPE,

am payments.amount%TYPE,

m\_id members.member\_id%TYPE,

new\_am payments.amount%TYPE) IS

BEGIN

update payments

set

amount=new\_am

where payment\_id=pid;

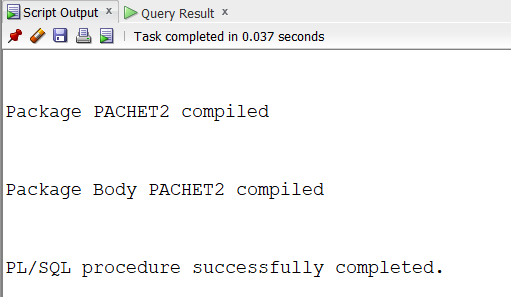
DBMS\_OUTPUT.PUT\_LINE('plata cu id-ul ' || pid ||' cu valoarea ' || am|| ' a fost modficata');

END modify\_payment;

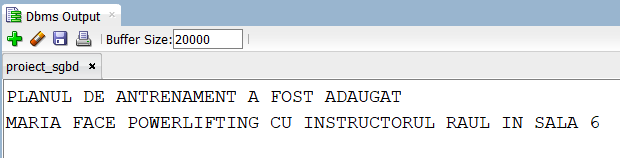
END pachet2;

/

drop package body pachet2;



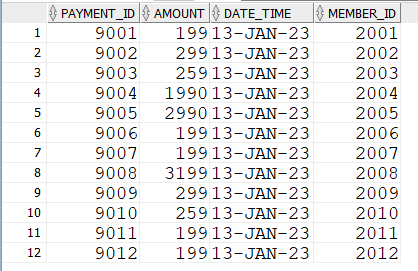
execute pachet2.add\_workout\_plan(6,7,10,2001);



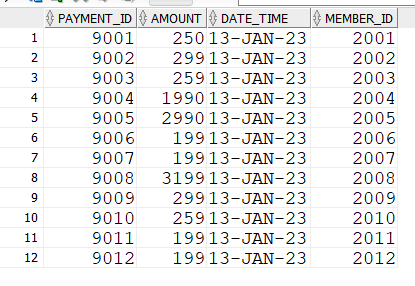
execute pachet2.modify\_payment(9001,199,2001,250);



inainte



Dupa



--ex14 refacut

-- facem un pachet care sa afiseze adresele salilor de fitness in care

-- se antreneaza membrii clubului

-- input : nume membru, nume instructor

-- tabel indexat pt a tine evidenta antrenamentelor (workout\_plan)

-- tabel imbricat pt a tine evidenta salilor (gyms)

-- 3 functii : get\_member\_id, get\_instructor\_id , get\_workout\_id

-- 2 proceduri : get\_workout\_plan, show\_gym

CREATE OR REPLACE PACKAGE pachet as

TYPE tip\_tabel\_workout\_plan IS TABLE OF workout\_plan%ROWTYPE INDEX BY PLS\_INTEGER;

tabel\_wk tip\_tabel\_workout\_plan;

TYPE tip\_tabel\_sali IS TABLE OF gyms.gym\_id%TYPE;

tabel\_gym tip\_tabel\_sali := tip\_tabel\_sali();

FUNCTION get\_member\_id (m\_name members.member\_name%TYPE)

RETURN members.member\_id%TYPE;

FUNCTION get\_instructor\_id (i\_name instructors.instructor\_name%TYPE)

RETURN instructors.instructor\_id%TYPE;

FUNCTION get\_workout\_id (wk\_name workout.workout\_name%TYPE)

RETURN workout.workout\_id%TYPE;

PROCEDURE get\_workout\_plan ( m\_name members.member\_name%TYPE,

i\_name instructors.instructor\_name%TYPE,

wk\_name workout.workout\_name%TYPE);

PROCEDURE show\_gym (g\_id gyms.gym\_id%TYPE);

end pachet;

/

CREATE OR REPLACE PACKAGE BODY pachet as

FUNCTION get\_member\_id (m\_name members.member\_name%TYPE)

RETURN members.member\_id%TYPE AS

v\_id members.member\_id%TYPE;

nr NUMBER(10):=0;

BEGIN

SELECT COUNT(MEMBER\_ID)

INTO NR

FROM MEMBERS

WHERE UPPER(MEMBER\_NAME)=UPPER(M\_NAME);

IF NR=0 THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA MEMBRII CU NUMELE DAT');

ELSIF NR>1 THEN

DBMS\_OUTPUT.PUT\_LINE('EXISTA MAI MULTI MEMBRII CU NUMELE DAT');

ELSE

SELECT MEMBER\_ID

INTO V\_ID

FROM MEMBERS

WHERE UPPER(MEMBER\_NAME)=UPPER(M\_NAME);

RETURN V\_ID;

END IF;

END get\_member\_id;

FUNCTION get\_instructor\_id (i\_name instructors.instructor\_name%TYPE)

RETURN instructors.instructor\_ID%TYPE AS

v\_id instructors.instructor\_ID%TYPE;

nr NUMBER(10):=0;

BEGIN

SELECT COUNT(instructor\_id)

INTO NR

FROM instructors

WHERE UPPER(instructor\_name)=UPPER(i\_name);

IF NR=0 THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA INSTRUCTORI CU NUMELE DAT');

ELSIF NR>1 THEN

DBMS\_OUTPUT.PUT\_LINE('EXISTA MAI MULTI INSTRUCTORI CU NUMELE DAT');

ELSE

SELECT instructor\_id

INTO V\_ID

FROM instructors

WHERE UPPER(instructor\_name)=UPPER(i\_name);

RETURN V\_ID;

END IF;

END get\_instructor\_id;

FUNCTION get\_workout\_id (wk\_name workout.workout\_name%TYPE)

RETURN workout.workout\_id%TYPE AS

v\_id workout.workout\_ID%TYPE;

nr NUMBER(10):=0;

BEGIN

SELECT COUNT(workout\_id)

INTO NR

FROM WORKOUT

WHERE UPPER(workout\_name)=UPPER(wk\_name);

IF NR=0 THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA WORKOUT CU NUMELE DAT');

ELSIF NR>1 THEN

DBMS\_OUTPUT.PUT\_LINE('EXISTA MAI MULTE WORKOUT-URI CU NUMELE DAT');

ELSE

SELECT workout\_id

INTO V\_ID

FROM workout

WHERE UPPER(workout\_name)=UPPER(wk\_name);

RETURN V\_ID;

END IF;

END GET\_WORKOUT\_ID;

PROCEDURE show\_gym (g\_id gyms.gym\_id%TYPE) AS

adresa gyms.adress%TYPE;

nr NUMBER(10):=0;

BEGIN

SELECT COUNT(\*)

INTO NR

FROM GYMS

WHERE GYM\_ID=G\_ID;

IF NR=0 THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA SALA');

ELSIF NR>1 THEN

DBMS\_OUTPUT.PUT\_LINE('EXISTA MAI MULTE SALI');

ELSE

SELECT ADRESS

INTO ADRESA

FROM GYMS

WHERE GYM\_ID=G\_ID;

DBMS\_OUTPUT.PUT\_LINE(ADRESA);

END IF;

END show\_gym;

PROCEDURE get\_workout\_plan ( m\_name members.member\_name%TYPE,

i\_name instructors.instructor\_name%TYPE,

wk\_name workout.workout\_name%TYPE) as

mid members.member\_id%TYPE;

iid instructors.instructor\_id%TYPE;

wkid workout.workout\_id%TYPE;

gid gyms.gym\_id%TYPE;

v\_wk workout\_plan%ROWTYPE;

BEGIN

mid:=get\_member\_id(m\_name);

IF mid!=0 then

iid:= get\_instructor\_id(i\_name);

wkid:=get\_workout\_id(wk\_name);

if iid!=0 and wkid!=0 then

SELECT \* BULK COLLECT INTO tabel\_wk

FROM workout\_plan

where instructor\_id=iid and workout\_id=wkid;

IF tabel\_wk.count>0 THEN

FOR i in tabel\_wk.FIRST..tabel\_wk.LAST LOOP

GID:=tabel\_wk(i).gym\_id;

tabel\_gym.extend;

tabel\_gym(i):=gid;

END LOOP;

IF tabel\_gym.count>0 THEN

FOR I IN tabel\_gym.FIRST..tabel\_gym.LAST LOOP

show\_gym(tabel\_gym(i));

END LOOP;

end if;

ELSE

DBMS\_OUTPUT.PUT\_LINE('NU SE POATE');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('NU SE POATE');

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('NU SE POATE');

END IF;

END GET\_WORKOUT\_PLAN;

end pachet;

/

drop package pachet;

--NO DATA FOUND MEMBRU : CORNEL MIHAELA SWIMMING

-- TOO MANY ROWS - MEMBRU : LORENA MIHAELA BODYBUILDING

--NO DATA FOUND -- INSTRUCTORI 'LUCA','IOANA','BODYBUILDING'

-- TOO MANY ROWS - INSTRUCTORI 'LUCA','RAUL','BODYBUILDING'

--NO DATA FOUND -- WORKOUT 'LILIANA','MIHAELA','JUMP'

-- TOO MANY ROWS - WORKOUT'LILIANA','MIHAELA','CARDIO'

--NU EXISTA PLANUL DE ANTRENAMENT 'LILIANA','ALEX','BODYBUILDING'

--SUCCESS 'LILIANA','ALEX','SWIMMING'

EXECUTE PACHET.GET\_WORKOUT\_PLAN('&P\_NUME\_MEMBRU','&P\_NUME\_INSTRUCTOR','&P\_NUME\_WORKOUT');