Gestionarea unei stupine

- Dogaru Mihail Dănuț

-grupa 232

Cuprins

[1.Prezentarea pe scurt a bazei de date 4](#_Toc155989413)

[2.Diagrama Entitate - Relație 4](#_Toc155989414)

[3.Diagrama conceptuala 5](#_Toc155989415)

[4.Definirea tabelelor și constrângerilor 5](#_Toc155989416)

[5.Inserarea informațiilor 9](#_Toc155989417)

[6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul. 20](#_Toc155989418)

[7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul. 25](#_Toc155989419)

[8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate. 28](#_Toc155989420)

[9. 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. 30](#_Toc155989421)

[10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. 33](#_Toc155989422)

[11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul. 36](#_Toc155989423)

[12. Definiți un trigger de tip LDD. Declanșați trigger-ul. 37](#_Toc155989424)

[13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului. 42](#_Toc155989425)

[14. 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). 53](#_Toc155989426)

# 1.Prezentarea pe scurt a bazei de date

Modelul de date va gestiona informații despre o stupină de multiplicare. Afacerea este gestionată de mai mulții apicultori licențiați ce își au stupii în mai multe locații pentru a acoperi o suprafață mai vastă, ajutând la producția de miere si diversitatea genetică. Fiecare apicultor poate merge în oricare dintre locație și poate verifica orice stup, stupii putând fi mutați dintr-o locație în alta.

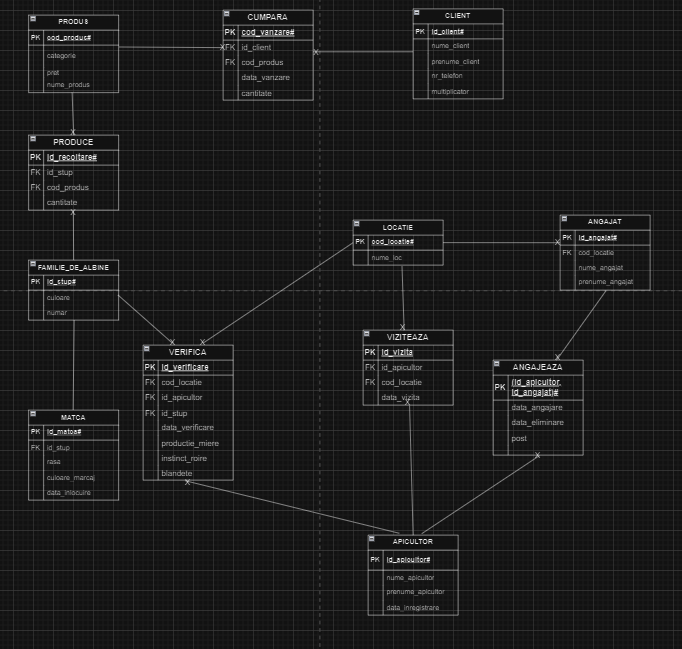
Apicultorii pot angaja lucrători pe diverse posturi, fiind asignați uneia dintre locații. Afacerea trebuie sa aibe întotdeauna un contabil, iar în fiecare locație cel puțin un paznic pentru ca stupii să nu fie furați. Angajații pot lucra o singură dată pentru firmă, concedierea fiind permanentă.

Pot exista stupi fără matcă, aceasta fiind adusă ulterior, dar poate fi și înlocuită cu una nouă, însă trebuie sa fie mereu într-o familie de albine. Mătcile sunt marcate respectând convenția de marcare,iar stupii sunt colorați pentru a ajuta la orientarea albinelor,dar sunt și numerotați pentru a facilita identificarea lor, însă au și un cod unic de înregistrare în cadrul județului în care este licențiată stupina.

Fiecare stup poate produce o varietate de produse apicole specifice consumului, dar și multiplicării și pot fi achiziționate de orice client.

# 2.Diagrama Entitate - Relație

# 3.Diagrama conceptuala



# 4.Definirea tabelelor și constrângerilor

create table APICULTOR(

id\_apicultor number(6) PRIMARY KEY,

nume\_apicultor varchar2(20) NOT NULL,

prenume\_apicultor varchar2(20) NOT NULL,

data\_inregistrare date NOT NULL

);

create table CLIENT(

id\_client number(6) PRIMARY KEY,

nume\_client varchar2(20) NOT NULL,

prenume\_client varchar2(20),

nr\_telefon varchar2(10) UNIQUE ,

multiplicator number(1)

);

create table PRODUS(

cod\_produs number(13) PRIMARY KEY,

categorie varchar2(20),

pret number(5,2),

nume\_produs varchar2(30) UNIQUE

);

create table LOCATIE(

cod\_locatie number(2) PRIMARY KEY,

nume\_loc varchar2(20) UNIQUE

);

create table ANGAJAT(

id\_angajat number(3) PRIMARY KEY,

cod\_locatie number(2) NOT NULL,

nume\_angajat varchar2(20),

prenume\_angajat varchar2(20)

);

create table FAMILIE\_DE\_ALBINE(

id\_stup varchar2(8) PRIMARY KEY,

culoare varchar2(15),

numar number(3)

);

create table MATCA(

id\_matca varchar2(3) PRIMARY KEY,

id\_stup varchar2(6),

rasa varchar2(20) CHECK( rasa = 'carpatina'),

culoare\_marcaj varchar2(1) CHECK(culoare\_marcaj = 'a' or culoare\_marcaj = 's' or culoare\_marcaj = 'r' or culoare\_marcaj = 'g' or culoare\_marcaj = 'v'),

data\_inlocuire date

);

create table CUMPARA(

cod\_vanzare number(8) PRIMARY KEY,

id\_client number(6),

cod\_produs number(13),

cantitate number(4),

data\_vanzare date

);

create table VIZITEAZA(

id\_vizita number(6) PRIMARY KEY,

id\_apicultor number(6),

cod\_locatie number(2),

data\_vizita date

);

create table ANGAJEAZA(

id\_apicultor number(6),

id\_angajat number(3),

data\_angajare date,

data\_eliminare date,

post varchar2(20),

salariu number(5) CHECK (salariu >1000)

);

create table PRODUCE(

id\_recoltare number(7) PRIMARY KEY,

id\_stup varchar2(6),

cod\_produs number(13),

cantitate number(4) CHECK(cantitate > 0)

);

create table VERIFICA(

id\_verificare number(7) PRIMARY KEY,

id\_stup varchar2(6),

id\_apicultor number(6),

cod\_locatie number(2),

productie\_miere number(1) CHECK(productie\_miere > 0 and productie\_miere < 6 ),

instinct\_roire number(1) CHECK(instinct\_roire > 0 and instinct\_roire < 6),

blandete number(1) CHECK(blandete > 0 and blandete < 6),

data\_verificare date

);

alter table ANGAJAT

add constraint fk\_cod\_locatie FOREIGN KEY(cod\_locatie)

REFERENCES LOCATIE(cod\_locatie);

alter table MATCA

add constraint fk\_id\_stup FOREIGN KEY(id\_stup)

references FAMILIE\_DE\_ALBINE(id\_stup);

alter table PRODUCE

add(constraint fk\_prod\_id\_stup FOREIGN KEY(id\_stup) references FAMILIE\_DE\_ALBINE(id\_stup),

constraint fk\_prod\_cod\_produs FOREIGN KEY(cod\_produs) references PRODUS(cod\_produs));

alter table VERIFICA

add(constraint fk\_ver\_id\_stup FOREIGN KEY(id\_stup) references FAMILIE\_DE\_ALBINE(id\_stup),

constraint fk\_ver\_id\_apicultor FOREIGN KEY(id\_apicultor) references APICULTOR(id\_apicultor),

constraint fk\_ver\_cod\_locatie FOREIGN KEY(cod\_locatie) references LOCATIE(cod\_locatie));

alter table ANGAJEAZA

add (constraint pk\_angajeaza PRIMARY KEY(id\_apicultor,id\_angajat),

constraint fk\_ang\_id\_apicultor FOREIGN KEY(id\_apicultor) references APICULTOR(id\_apicultor),

constraint fk\_ang\_id\_angajat FOREIGN KEY(id\_angajat) references ANGAJAT(id\_angajat));

alter table VIZITEAZA

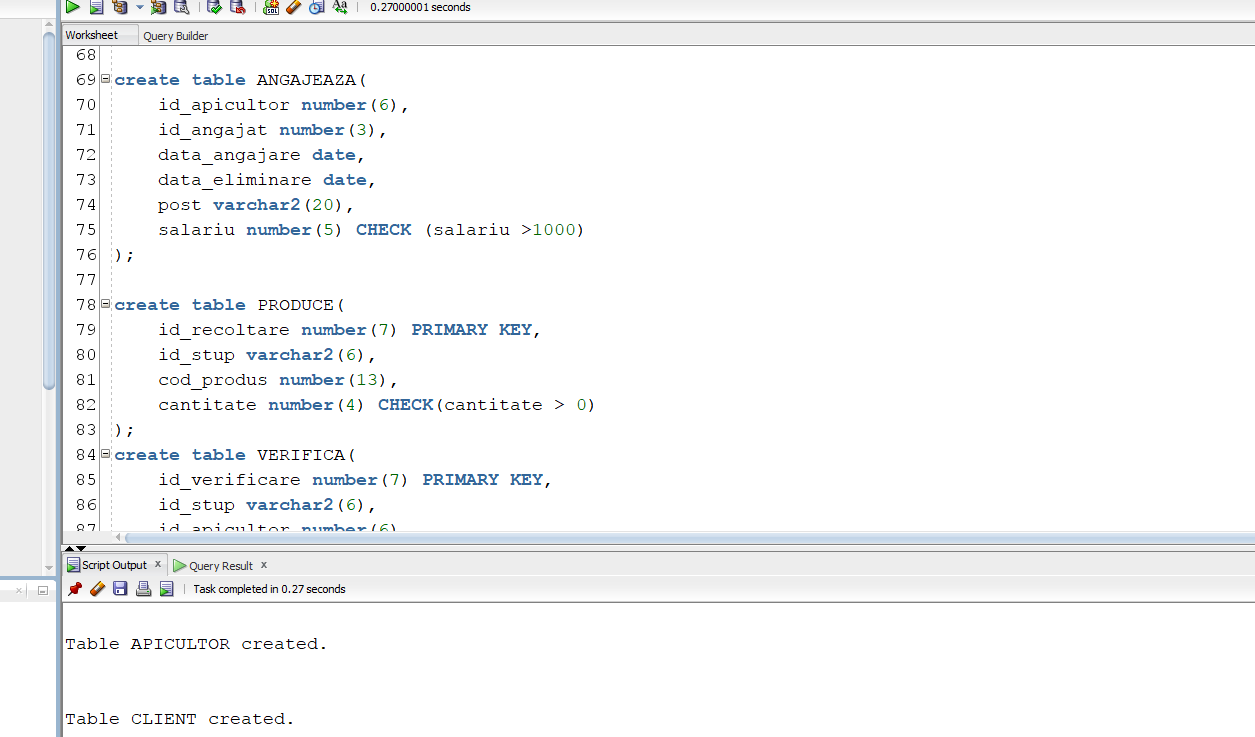
add(constraint fk\_viz\_id\_apicultor FOREIGN KEY(id\_apicultor) references APICULTOR(id\_apicultor),

constraint fk\_viz\_cod\_locatie FOREIGN KEY (cod\_locatie) references LOCATIE(cod\_locatie));

alter table CUMPARA

add( constraint fk\_cump\_id\_client FOREIGN KEY(id\_client) references CLIENT(id\_client),

constraint fk\_cump\_cod\_produs FOREIGN KEY(cod\_produs) references PRODUS(cod\_produs));



# 5.Inserarea informațiilor

create SEQUENCE index\_ang

START WITH 100

INCREMENT BY 1

MAXVALUE 999

NOCACHE

NOCYCLE;

create SEQUENCE cl\_index

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

create SEQUENCE index\_loc

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

create SEQUENCE index\_vanzare

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

create SEQUENCE index\_vizita

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

create SEQUENCE index\_recoltare

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

create SEQUENCE index\_verificare

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;

insert into APICULTOR values (141042, 'Dogaru','Mihai',to\_date('24.10.2009','dd.mm.yyyy'));

insert into APICULTOR values (104921 , 'Dogaru','Sorin',to\_date('12.04.2003','dd.mm.yyyy'));

insert into APICULTOR values (947821 , 'Mircea','Razvan',to\_date('05.02.2013','dd.mm.yyyy'));

insert into APICULTOR values (190391 , 'Alexandru','Andrei',to\_date('10.03.2023','dd.mm.yyyy'));

insert into APICULTOR values (194821, 'Papusoi','Rares',to\_date('01.02.2010','dd.mm.yyyy'));

--select \* from APICULTOR;

insert into CLIENT values(cl\_index.nextval,'Popescu','Andrei','0728109930',0);

insert into CLIENT values(cl\_index.nextval,'Ionescu','Razvan','0768269121',0);

insert into CLIENT values(cl\_index.nextval,'Mircea','Andrei','0312765980',0);

insert into CLIENT values(cl\_index.nextval,'Popescu','Alexandru','0772541120',0);

insert into CLIENT values(cl\_index.nextval,'Bucurie','Bogdan','0796312585',0);

insert into CLIENT values(cl\_index.nextval,'Iuliu','Daniel','0787654312',0);

insert into CLIENT values(cl\_index.nextval,'Anghel','Rares','0798765432',0);

insert into CLIENT values(cl\_index.nextval,'Paun','Andrei','0712311111',0);

insert into CLIENT values(cl\_index.nextval,'Ghergu','Cezar','0387645674',0);

insert into CLIENT values(cl\_index.nextval,'Grigore','Vasile','0700122332',0);

insert into client values (cl\_index.nextval, 'Coroama','Cristian',NULL,0);

insert into client values (cl\_index.nextval, 'Tirila','Patric',NULL,0);

insert into client values (cl\_index.nextval, 'Gavrila','Alexandru',NULL,0);

insert into client values (cl\_index.nextval, 'Codarcea','Alexandru',NULL,0);

--select \* from CLIENT;

insert into PRODUS values(12345678901,'consumabil',20.5 ,'miere de poliflora');

insert into PRODUS values(12345678902,'consumabil',35.5 ,'miere de salcam');

insert into PRODUS values(12345678903,'consumabil',25.5 ,'miere de rapita');

insert into PRODUS values(63451424121,'biologic', 80,'matca');

insert into PRODUS values(11234121341,'biologic',400 ,'roi');

insert into PRODUS values(51231411123,NULL,20 ,'tinctura de propolis');

insert into PRODUS values(99014141212,'consumabil',25,'laptisor de matca');

insert into produs values(14252641314,'biologic','25','botca');

insert into produs values('12345432112','consumbabil',50,'polen');

--select \* from PRODUS;

insert into LOCATIE values(index\_loc.nextval,'padure ciornuleasa');

insert into LOCATIE values(index\_loc.nextval,'mamaia');

insert into LOCATIE values(index\_loc.nextval,'camp');

insert into LOCATIE values(index\_loc.nextval,'acasa');

insert into LOCATIE values(index\_loc.nextval,'padure curcani');

--select \* from LOCATIE;

insert into ANGAJAT values(index\_ang.nextval,1,'Gica',NULL);

insert into ANGAJAT values(index\_ang.nextval,2,'Bora','Dragos');

insert into ANGAJAT values(index\_ang.nextval,3,'Coxescu','Marian');

insert into ANGAJAT values(index\_ang.nextval,4,'Georgel','Andrei');

insert into ANGAJAT values(index\_ang.nextval,5,'Hoara','Ionut');

insert into ANGAJAT values(index\_ang.nextval,4,'Ene','Cristi');

insert into ANGAJAT values(index\_ang.nextval,3,'Jac',NULL);

insert into ANGAJAT values(index\_ang.nextval,4,'Bobita',NULL);

insert into ANGAJAT values(index\_ang.nextval,1,'Cristi','Cristi');

insert into ANGAJAT values(index\_ang.nextval,4,'Ionut','Ionut');

--select \* from ANGAJAT;

--loc1

insert into FAMILIE\_DE\_ALBINE values('CL0001','verde',12);

insert into FAMILIE\_DE\_ALBINE values('CL0012','rosu',13);

insert into FAMILIE\_DE\_ALBINE values('CL0043','galben',22);

insert into FAMILIE\_DE\_ALBINE values('CL0024','albastru',2);

insert into FAMILIE\_DE\_ALBINE values('CL0110','alb',10);

insert into FAMILIE\_DE\_ALBINE values('CL0021','verde',17);

insert into FAMILIE\_DE\_ALBINE values('CL0031','albastru',56);

insert into FAMILIE\_DE\_ALBINE values('CL0041','rosu',101);

insert into FAMILIE\_DE\_ALBINE values('CL0054','alb',345);

insert into FAMILIE\_DE\_ALBINE values('CL0087','rosu',19);

--loc2

insert into FAMILIE\_DE\_ALBINE values('CL0056','verde',1);

insert into FAMILIE\_DE\_ALBINE values('CL0023','rosu',77);

insert into FAMILIE\_DE\_ALBINE values('CL0112','alb',102);

insert into FAMILIE\_DE\_ALBINE values('CL0101','alb',100);

insert into FAMILIE\_DE\_ALBINE values('CL0201','albastru',71);

--loc3

insert into FAMILIE\_DE\_ALBINE values('CL0090','verde',321);

insert into FAMILIE\_DE\_ALBINE values('CL0009','rosu',118);

insert into FAMILIE\_DE\_ALBINE values('CL1111','alb',512);

insert into FAMILIE\_DE\_ALBINE values('CL3101','alb',24);

insert into FAMILIE\_DE\_ALBINE values('CL4201','albastru',171);

insert into FAMILIE\_DE\_ALBINE values('CL5656','verde',906);

insert into FAMILIE\_DE\_ALBINE values('CL1002','rosu',471);

insert into FAMILIE\_DE\_ALBINE values('CL3172','alb',903);

insert into FAMILIE\_DE\_ALBINE values('CL3501','alb',156);

insert into FAMILIE\_DE\_ALBINE values('CL9109','albastru',781);

insert into FAMILIE\_DE\_ALBINE values('CL7456','verde',822);

insert into FAMILIE\_DE\_ALBINE values('CL1223','rosu',773);

insert into FAMILIE\_DE\_ALBINE values('CL7410','alb',404);

insert into FAMILIE\_DE\_ALBINE values('CL0381','alb',828);

insert into FAMILIE\_DE\_ALBINE values('CL1411','albastru',971);

insert into FAMILIE\_DE\_ALBINE values('CL8806','verde',991);

insert into FAMILIE\_DE\_ALBINE values('CL7323','rosu',797);

insert into FAMILIE\_DE\_ALBINE values('CL6312','alb',717);

insert into FAMILIE\_DE\_ALBINE values('CL6221','alb',530);

insert into FAMILIE\_DE\_ALBINE values('CL9301','albastru',978);

--loc4

insert into FAMILIE\_DE\_ALBINE values('CL9806','verde',900);

insert into FAMILIE\_DE\_ALBINE values('CL8323','rosu',700);

insert into FAMILIE\_DE\_ALBINE values('CL7312','alb',117);

insert into FAMILIE\_DE\_ALBINE values('CL7221','alb',500);

insert into FAMILIE\_DE\_ALBINE values('CL1301','albastru',970);

insert into FAMILIE\_DE\_ALBINE values('CL9807','rosu',904);

insert into FAMILIE\_DE\_ALBINE values('CL8324','rosu',710);

insert into FAMILIE\_DE\_ALBINE values('CL7313','alb',515);

insert into FAMILIE\_DE\_ALBINE values('CL7022','verde',487);

insert into FAMILIE\_DE\_ALBINE values('CL5302','bej',877);

--loc5

insert into FAMILIE\_DE\_ALBINE values('CL8807','verde',999);

insert into FAMILIE\_DE\_ALBINE values('CL9423','rosu',799);

insert into FAMILIE\_DE\_ALBINE values('CL8412','alb',199);

insert into FAMILIE\_DE\_ALBINE values('CL8321','alb',599);

insert into FAMILIE\_DE\_ALBINE values('CL2401','albastru',222);

insert into FAMILIE\_DE\_ALBINE values('CL8907','rosu',333);

insert into FAMILIE\_DE\_ALBINE values('CL1824','rosu',444);

insert into FAMILIE\_DE\_ALBINE values('CL7553','alb',151);

insert into FAMILIE\_DE\_ALBINE values('CL7442','verde',666);

insert into FAMILIE\_DE\_ALBINE values('CL5444','alb',777);

--select \* from FAMILIE\_DE\_ALBINE;

insert into MATCA values('AA','CL0001','carpatina','a',NULL);

insert into MATCA values('AB','CL0012','carpatina','a',NULL);

insert into MATCA values('AC','CL0110','carpatina','g',NULL);

insert into MATCA values('AD','CL0021','carpatina','g',NULL);

insert into MATCA values('AR','CL0031','carpatina','v',NULL);

insert into MATCA values('AT','CL0041','carpatina','r',NULL);

insert into MATCA values('AY','CL0087','carpatina','r',NULL);

insert into MATCA values('AU','CL0023','carpatina','s',NULL);

insert into MATCA values('AI','CL0056','carpatina','a',NULL);

insert into MATCA values('AO','CL0101','carpatina','g',NULL);

insert into MATCA values('BT','CL0090','carpatina','r',NULL);

insert into MATCA values('BY','CL0009','carpatina','s',NULL);

insert into MATCA values('BU','CL3101','carpatina','s',NULL);

insert into MATCA values('BI','CL5656','carpatina','a',NULL);

insert into MATCA values('BR','CL9806','carpatina','v',NULL);

insert into MATCA values('BE','CL5444','carpatina','a',NULL);

insert into MATCA values('CY','CL1223','carpatina','r',NULL);

insert into MATCA values('CU','CL0381','carpatina','v',NULL);

insert into MATCA values('CI','CL8806','carpatina','a',NULL);

insert into MATCA values('WO','CL7323','carpatina','g',NULL);

insert into MATCA values('EE','CL7022','carpatina','v',NULL);

insert into MATCA values('QW','CL9423','carpatina','a',NULL);

--select \* from MATCA

insert into cumpara values(index\_vanzare.nextval, 1,12345678901,2,to\_date('23.01.2019','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 2,12345678901,1,to\_date('14.11.2019','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 2,12345678903,4,to\_date('20.01.2017','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 1,12345678902 ,4,to\_date('10.04.2020','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 3,12345678901,7,to\_date('27.04.2016','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 1,99014141212,10,to\_date('13.07.2018','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 10,12345678901,3,to\_date('20.02.2022','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 5,63451424121,23,to\_date('15.11.2023','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 3,11234121341,5,to\_date('20.05.2020','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 4,14252641314,10,to\_date('23.01.2019','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 9,12345678901,1,to\_date('21.11.2009','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 8,63451424121,12,to\_date('24.05.2018','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 7,12345678901,11,to\_date('20.03.2020','dd.mm.yyyy'));

insert into cumpara values(index\_vanzare.nextval, 6,11234121341,2,to\_date('03.01.2023','dd.mm.yyyy'));

--select \* from CUMPARA;

insert into viziteaza values(index\_vizita.nextval,141042,1,to\_date('15.10.2020','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,104921,2,to\_date('16.11.2019','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,947821,3,to\_date('19.05.2022','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,947821,4,to\_date('02.02.2020','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,194821,5,to\_date('01.12.2009','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,104921,2,to\_date('26.04.2010','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,104921,3,to\_date('30.11.2019','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,104921,1,to\_date('15.06.2022','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,141042,5,to\_date('14.05.2018','dd.mm.yyyy'));

insert into viziteaza values(index\_vizita.nextval,947821,5,to\_date('15.10.2019','dd.mm.yyyy'));

--select \* from VIZITEAZA;

insert into angajeaza values (141042,100 ,to\_date('04.07.2017','dd.mm.yyyy'),NULL,'paznic','1700');

insert into angajeaza values (104921,101 ,to\_date('12.03.2018','dd.mm.yyyy'),NULL,'paznic','1650');

insert into angajeaza values (141042,102 ,to\_date('14.04.2020','dd.mm.yyyy'),to\_date('11.12.2009','dd.mm.yyyy'),'sofer','1400');

insert into angajeaza values (141042,103 ,to\_date('10.12.2009','dd.mm.yyyy'),to\_date('11.12.2009','dd.mm.yyyy'),'asistent','1800');

insert into angajeaza values (104921,104 ,to\_date('10.12.2009','dd.mm.yyyy'),NULL,'paznic','1700');

insert into angajeaza values (104921,105 ,to\_date('15.05.2010','dd.mm.yyyy'),NULL,'contabil','2500');

insert into angajeaza values (194821,106 ,to\_date('14.01.2019','dd.mm.yyyy'),NULL,'paznic','1800');

insert into angajeaza values (104921,107 ,to\_date('10.12.2009','dd.mm.yyyy'),NULL,'paznic','1700');

insert into angajeaza values (194821,108 ,to\_date('24.09.2011','dd.mm.yyyy'),NULL,'asistent','1800');

insert into angajeaza values (104921,109 ,to\_date('07.04.2020','dd.mm.yyyy'),NULL,'ambalator','1700');

--select \* from ANGAJEAZA;

insert into produce values(index\_recoltare.nextval,'CL8807',12345678901,10);

insert into produce values(index\_recoltare.nextval,'CL9423',12345678902,35);

insert into produce values(index\_recoltare.nextval,'CL8412',12345678903,20);

insert into produce values(index\_recoltare.nextval,'CL8321',99014141212,10);

insert into produce values(index\_recoltare.nextval,'CL2401',12345432112,1);

insert into produce values(index\_recoltare.nextval,'CL8907',12345678901,10);

insert into produce values(index\_recoltare.nextval,'CL1824',12345678901,14);

insert into produce values(index\_recoltare.nextval,'CL0054',12345678901,26);

insert into produce values(index\_recoltare.nextval,'CL0056',12345678901,21);

insert into produce values(index\_recoltare.nextval,'CL0101',12345678901,20);

insert into produce values(index\_recoltare.nextval,'CL0201',12345678902,23);

insert into produce values(index\_recoltare.nextval,'CL3101',12345678901,13);

insert into produce values(index\_recoltare.nextval,'CL4201',12345678901,24);

insert into produce values(index\_recoltare.nextval,'CL5656',12345678902,17);

insert into produce values(index\_recoltare.nextval,'CL9109',12345678901,35);

insert into produce values(index\_recoltare.nextval,'CL1301',12345678902,25);

insert into produce values(index\_recoltare.nextval,'CL7312',63451424121,1);

--select \* from PRODUCE;

insert into verifica values(index\_verificare.nextval,'CL8907',104921,1,4,2,2,to\_date('10.09.2023','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL4201',141042,1,3,3,5,to\_date('08.07.2022','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL3172',190391,2,1,4,5,to\_date('02.09.2019','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL1411',190391,2,5,5,5,to\_date('03.04.2009','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL7312',194821,3,4,4,3,to\_date('04.07.2010','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL1301',947821,4,2,3,4,to\_date('02.06.2012','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL7221',947821,3,1,4,5,to\_date('11.09.2014','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL7312',104921,3,4,4,5,to\_date('12.07.2014','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL0001',104921,5,4,4,2,to\_date('14.05.2016','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL8412',141042,5,1,2,3,to\_date('16.04.2015','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL7221',141042,1,4,1,1,to\_date('17.06.2020','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL7312',947821,4,2,3,3,to\_date('10.03.2021','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL0001',141042,4,4,4,2,to\_date('11.07.2022','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL8412',141042,4,5,5,5,to\_date('26.08.2023','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL3172',194821,2,3,2,4,to\_date('30.06.2023','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL1411',190391,4,2,3,4,to\_date('10.04.2018','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL7312',194821,4,2,3,3,to\_date('18.06.2017','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL1301',104921,4,1,2,2,to\_date('25.07.2014','dd.mm.yyyy'));

insert into verifica values(index\_verificare.nextval,'CL7221',947821,2,5,2,5,to\_date('27.08.2013','dd.mm.yyyy'));

--select \* from VERIFICA;

# 6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Se dă un număr k care reprezintă primele k luni ale anului și un număr val care reprezintă o suma de bani pe o singură plată

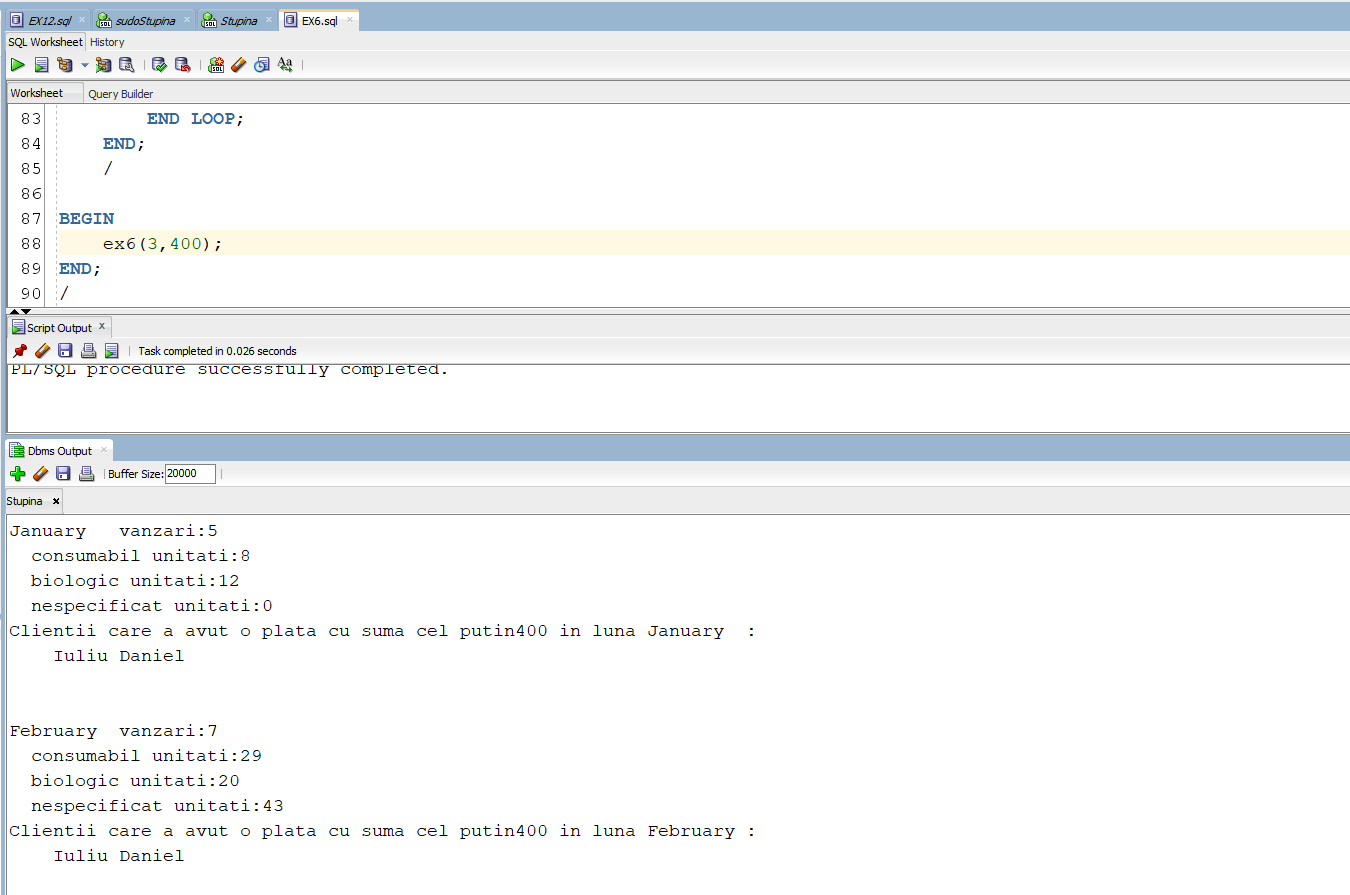
Să se afișeze pentru primele k luni:

-> cate plăți au fost realizate

-> cate unități din fiecare categorie au fost achiziționate

-> care au fost clienții care în luna i-au cheltuit cel puțin val

unități monetare pe o singură plată, unde i <-[1,k]



CREATE OR REPLACE PROCEDURE ex6

(k NUMBER, val NUMBER)

IS

--va stoca lunile din an in care s-au facut vanzari

TYPE vector\_luni IS VARRAY(12) OF NUMBER;

--va stoca categoriile de produse

TYPE tablou\_indexat\_categorii IS TABLE OF PLS\_INTEGER INDEX BY produs.categorie%TYPE;

--va stoca clientii care au cheltuit ... in luna respectiva

TYPE tablou\_imbricat\_clienti IS TABLE OF VARCHAR2(50);

t\_luni vector\_luni := vector\_luni();

t\_categorii tablou\_indexat\_categorii;

t\_clienti tablou\_imbricat\_clienti;

v\_aux NUMBER;

v\_aux\_nume\_luna VARCHAR2(20);

v\_aux\_nume\_client VARCHAR2(40);

BEGIN

-- ia numarul de vanzari din fiecare luna

FOR i IN 1..k LOOP

--facem un tabel nou pentru fiecare luna

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

t\_luni.extend;

select count(extract(month from data\_vanzare))

into t\_luni(i)

from cumpara

where i = extract(month from data\_vanzare);

--converteste din int in numele lunii

SELECT TO\_CHAR(TO\_DATE(i, 'MM'), 'Month')

into v\_aux\_nume\_luna

from dual;

DBMS\_OUTPUT.PUT\_LINE(v\_aux\_nume\_luna||' vanzari:'||t\_luni(i));

-- cate vanzari au fost din fiecare categorie

FOR v\_categ IN (SELECT DISTINCT nvl(categorie, 'nespecificat') AS category FROM produs) LOOP

SELECT nvl(SUM(c.cantitate),0)

INTO t\_categorii(v\_categ.category)

FROM cumpara c, produs p

WHERE p.cod\_produs = c.cod\_produs

AND nvl(p.categorie, 'nespecificat') = v\_categ.category

AND EXTRACT(MONTH FROM c.data\_vanzare) = i ;

DBMS\_OUTPUT.PUT\_LINE(' '||v\_categ.category||' unitati:'||t\_categorii(v\_categ.category));

END LOOP;

--ia clientii care au facut macar o plata mai mare decat parametru val in acea luna

SELECT DISTINCT cl.nume\_client||' '||cl.prenume\_client

BULK COLLECT INTO t\_clienti

FROM cumpara c, produs p, client cl

WHERE c.cod\_produs = p.cod\_produs

AND cl.id\_client = c.id\_client

AND EXTRACT(MONTH FROM c.data\_vanzare) = i

AND p.pret \* c.cantitate >= val;

IF t\_clienti.COUNT = 0

THEN DBMS\_OUTPUT.PUT\_LINE('Nu a fost niciun client cu o plata asa mare');

ELSE DBMS\_OUTPUT.PUT\_LINE('Clientii care a avut o plata cu suma cel putin'||val||' in luna '||v\_aux\_nume\_luna||':');

FOR i IN 1..t\_clienti.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(' '||t\_clienti(i));

END LOOP;

END IF;

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

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

END LOOP;

END;

/

BEGIN

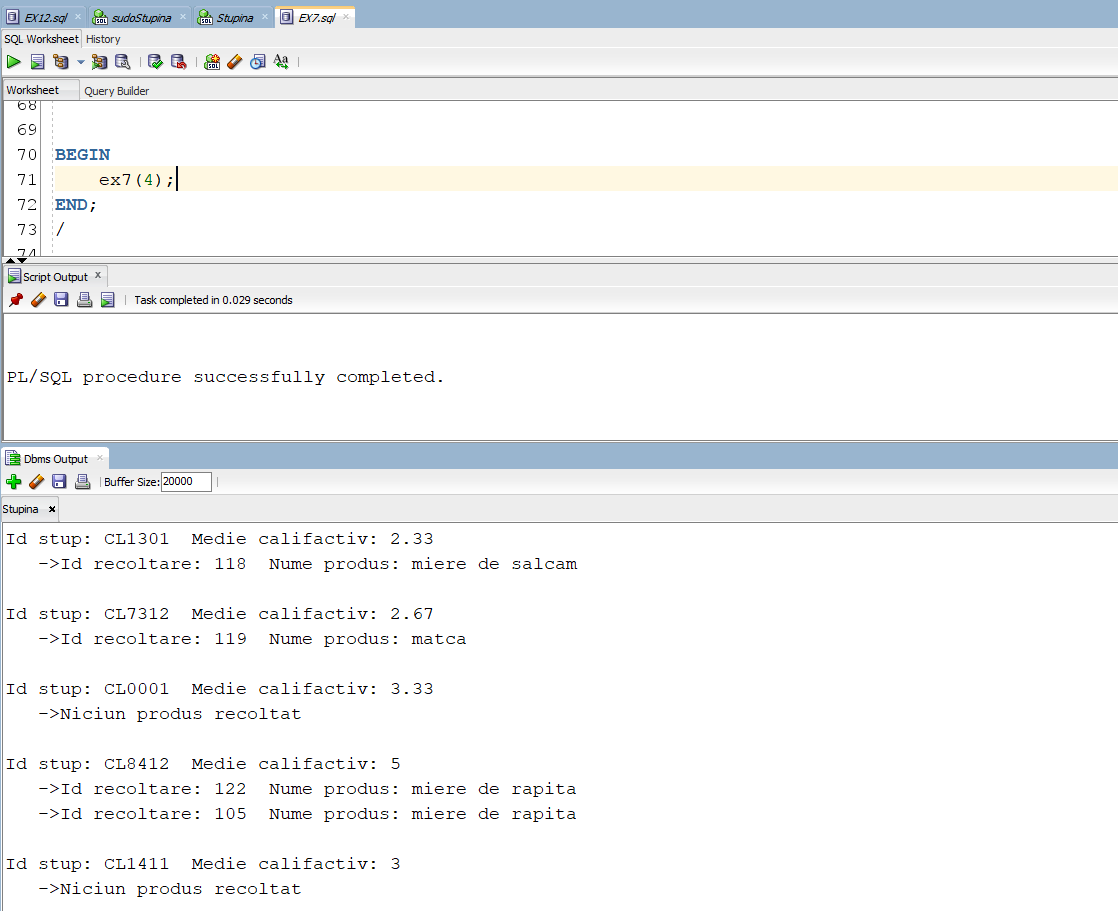
ex6(3,400);

END;

/

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

Să se afișeze nota stupilor verificați în anul curent din locația par\_loc si produsele recoltate de la aceștia.



CREATE OR REPLACE PROCEDURE ex7

(par\_loc locatie.cod\_locatie%TYPE DEFAULT 1)

IS

TYPE refcursor IS REF CURSOR;

CURSOR c (par\_cursor locatie.cod\_locatie%TYPE) IS

-- id media mediei calificativelor

SELECT l.id\_stup ,avg((blandete + instinct\_roire + productie\_miere)/3),

--intoarce produsele recoltate de la stupii respectivi

CURSOR(select p.id\_recoltare,pr.nume\_produs

from familie\_de\_albine f, produce p,produs pr

where f.id\_stup = p.id\_stup

and l.id\_stup = f.id\_stup

and pr.cod\_produs = p.cod\_produs)

FROM verifica l

--verifica locatia potrivita

WHERE par\_cursor = l.cod\_locatie

--verifica anul potrivit

AND EXTRACT(YEAR FROM l.data\_verificare)

= EXTRACT(YEAR FROM sysdate)

GROUP BY l.id\_stup;

v\_cursor refcursor;

v\_id\_stup verifica.id\_stup%TYPE;

v\_medie NUMBER;

v\_id\_rec produce.id\_recoltare%TYPE;

v\_nume\_prod produs.nume\_produs%TYPE;

v\_cnt\_lines NUMBER;

BEGIN

OPEN c(par\_loc);

LOOP

FETCH c INTO v\_id\_stup, v\_medie, v\_cursor;

EXIT WHEN c%NOTFOUND;

v\_cnt\_lines := 0;

DBMS\_OUTPUT.PUT\_LINE('Id stup: '||v\_id\_stup||' Medie califactiv: '||round(v\_medie,2));

LOOP

FETCH v\_cursor INTO v\_id\_rec, v\_nume\_prod;

EXIT WHEN v\_cursor%NOTFOUND;

v\_cnt\_lines := 1;

DBMS\_OUTPUT.PUT\_LINE(' ->Id recoltare: '||v\_id\_rec||' Nume produs: '||v\_nume\_prod);

END LOOP;

--verifica daca cursorul a returnat macar o linie pentru a afisa mesajul

IF v\_cnt\_lines = 0 THEN

DBMS\_OUTPUT.PUT\_LINE(' ->Niciun produs recoltat');

END IF;

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

END LOOP;

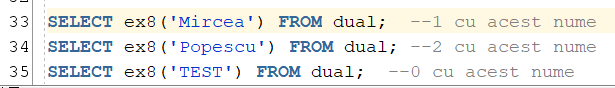
CLOSE c;

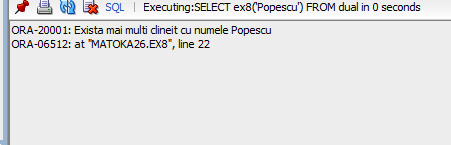
END;

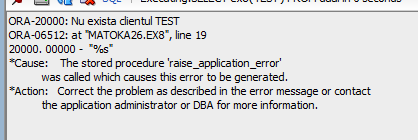
/

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

Pentru un client dat după nume, să se returneze suma de bani plătită în total pe cumpărături.







CREATE OR REPLACE FUNCTION ex8

(par\_nume client.nume\_client%TYPE)

RETURN NUMBER IS

v\_result NUMBER;

v\_id client.id\_client%TYPE;

BEGIN

-- am folosit clauza de group by pentru a rezolva situatiile conflictuale de nume

SELECT c.id\_client, SUM(c.cantitate \* p.pret)

INTO v\_id,v\_result

FROM cumpara c, client cl, produs p

WHERE cl.id\_client = c.id\_client

AND INITCAP(cl.nume\_client) = INITCAP(par\_nume)

AND c.cod\_produs = p.cod\_produs

group by c.id\_client;

RETURN v\_result;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000,

'Nu exista clientul '||par\_nume);

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001,

'Exista mai multi clineit cu numele '||par\_nume);

WHEN OTHERS THEN

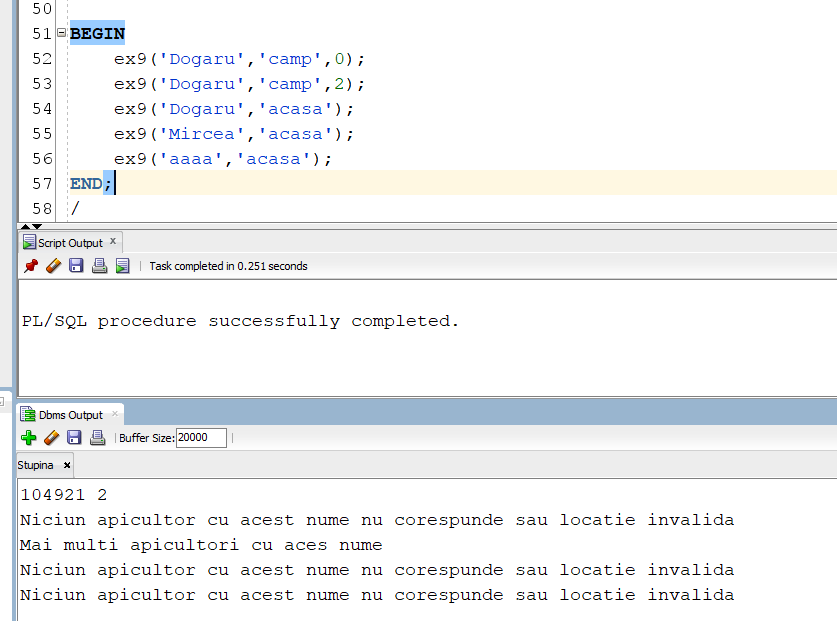
RAISE\_APPLICATION\_ERROR(-20002,'Alta eroare!');

END ex8;

/

# 9. 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.

Se dau un nume de apicultor, un nume de locație si un număr K. Să se afișeze id-ul apicultorului si câți angajați care lucrează în prezent a angajat în acea locație, dacă acesta(apicultorul) a vizitat mai mult de K locații.



CREATE OR REPLACE PROCEDURE ex9

(par\_nume apicultor.nume\_apicultor%TYPE,

par\_locatie locatie.nume\_loc%TYPE,

par\_k\_vizite NUMBER DEFAULT 0)

IS

v\_id\_apic apicultor.id\_apicultor%TYPE;

v\_cnt\_ang NUMBER;

v\_cnt\_viz NUMBER;

BEGIN

--cati angajati a angajat apicultorul in acea locatie si inca lucreaza

SELECT ap.id\_apicultor,count(a.id\_angajat), count(id\_vizita)

INTO v\_id\_apic, v\_cnt\_ang,v\_cnt\_viz

FROM apicultor ap, locatie l, angajeaza an,angajat a, viziteaza v

WHERE

--cauta apicultor dupa nume

ap.nume\_apicultor = Initcap(par\_nume)

--cauta locatia dupa nume

AND l.nume\_loc = par\_locatie

-- join pentru angajati pe locatie

AND a.id\_angajat = an.id\_angajat

AND a.cod\_locatie = l.cod\_locatie

--join pt count(id\_angajat) ->cati a angajat in acea locatie si inca lucreaza

AND ap.id\_apicultor = an.id\_apicultor

AND an.data\_eliminare is null

--join uri pentru count (id\_vizita)

AND v.id\_apicultor = ap.id\_apicultor

AND v.cod\_locatie = l.cod\_locatie

group by ap.id\_apicultor

having count(id\_vizita) > par\_k\_vizite;

DBMS\_OUTPUT.PUT\_LINE(v\_id\_apic||' '||v\_cnt\_ang);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Niciun apicultor cu acest nume nu corespunde sau locatie invalida');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Mai multi apicultori cu aces nume');

WHEN OTHERS THEN

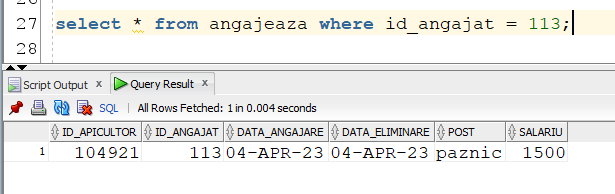
DBMS\_OUTPUT.PUT\_LINE('Alta eroare');

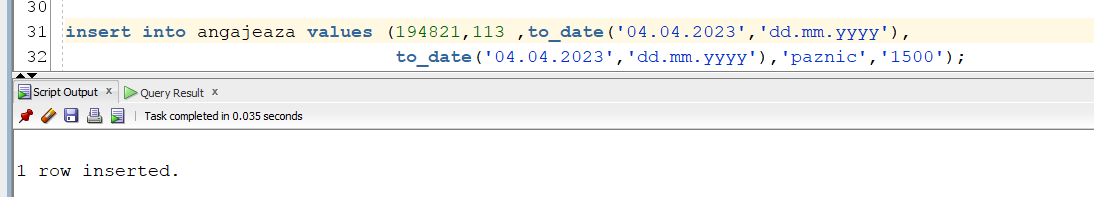
END;

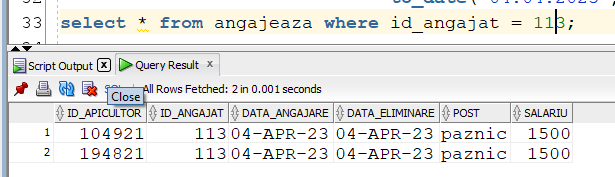
/

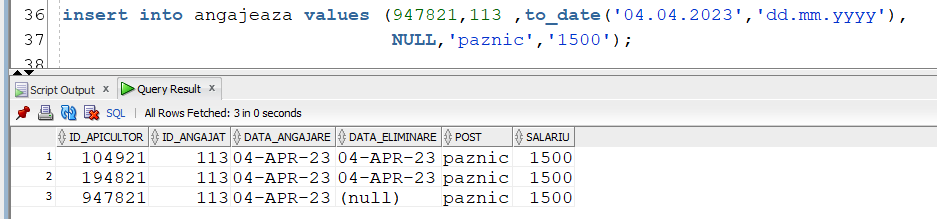
# 10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

Să se evite situația în care un angajat să fie angajat de 2ori in prezent, daca data\_eliminare = null ==> lucrează în prezent (nu a fost înca concediat)



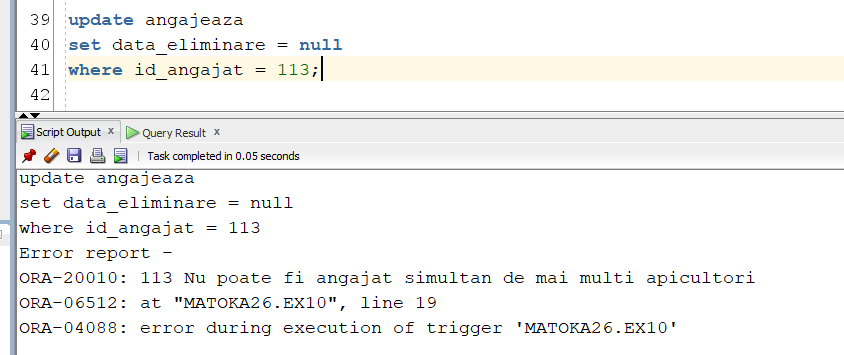




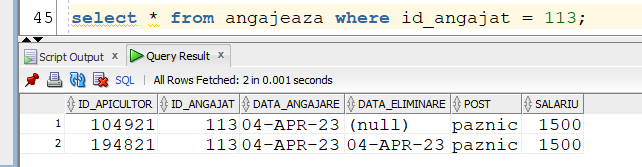


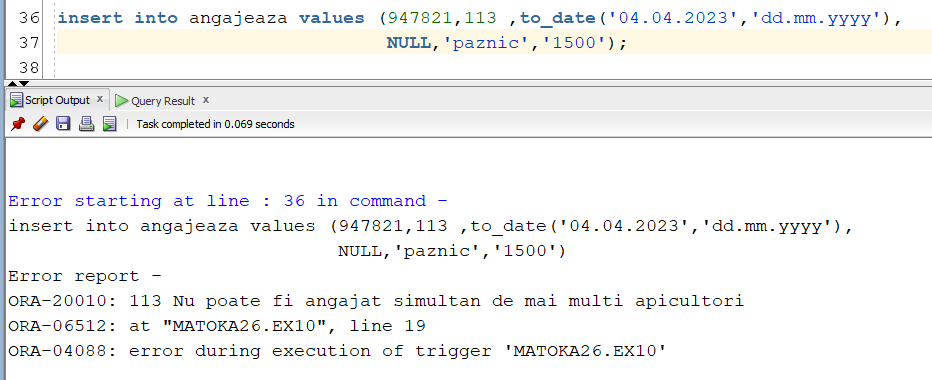
🡪Ceea ce este OK deoarece figurează ca angajat si concediat de 2ori

🡪Dacă încercăm să modificăm prin UPDATE o dată de concediere în NULL când există deja măcar una



🡪Dacă încercăm să inserăm o noua linie cu NULL când deja există una





CREATE OR REPLACE TRIGGER ex10

AFTER INSERT OR UPDATE ON angajeaza

DECLARE

--pentru fiecare angajat verifica de cate ori apare are data concedierii null

--semnificand ca lucreaza in prezent, aceasta poate fi 0(nu mai lucreaza deloc)

--sau 1(lucreaza in prezent)

CURSOR c is(select id\_angajat,sum(nvl2(data\_eliminare,0,1))

from angajeaza

group by id\_angajat);

v\_id\_angajat angajeaza.id\_angajat%TYPE;

v\_count PLS\_INTEGER;

BEGIN

OPEN c;

LOOP

FETCH c INTO v\_id\_angajat, v\_count;

EXIT WHEN c%NOTFOUND;

--daca se gaseste un angajat cu mai mult de 1 angajare

IF v\_count > 1 THEN

RAISE\_APPLICATION\_ERROR(-20010,v\_id\_angajat||' Nu poate fi angajat simultan de mai multi apicultori');

END IF;

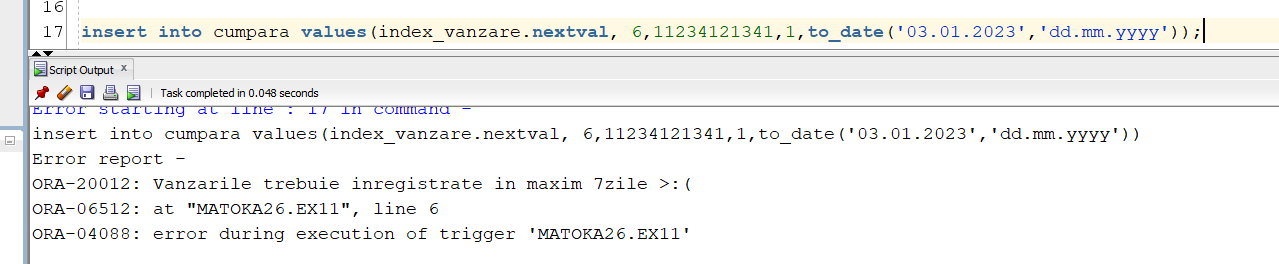
END LOOP;

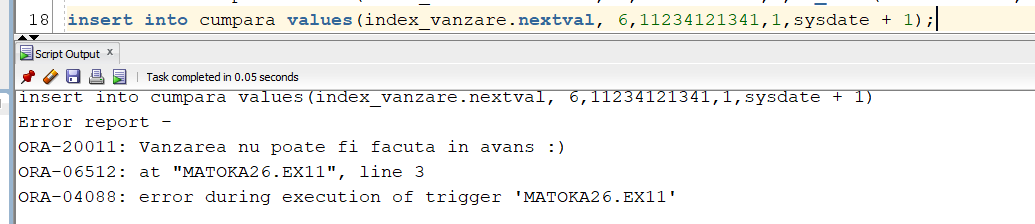
END;

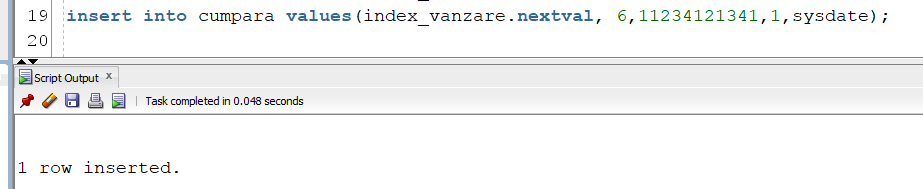
/

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

Să nu se permită vânzări pe date viitoare sau mai vechi de 1 săptămână







CREATE OR REPLACE TRIGGER ex11

BEFORE INSERT OR UPDATE ON cumpara

FOR EACH ROW

BEGIN

IF :NEW.data\_vanzare > sysdate THEN

RAISE\_APPLICATION\_ERROR(-20011, 'Vanzarea nu poate fi facuta in avans :)');

END IF;

IF :new.data\_vanzare < sysdate - 7 THEN

RAISE\_APPLICATION\_ERROR(-20012, 'Vanzarile trebuie inregistrate in maxim 7zile >:(');

END IF;

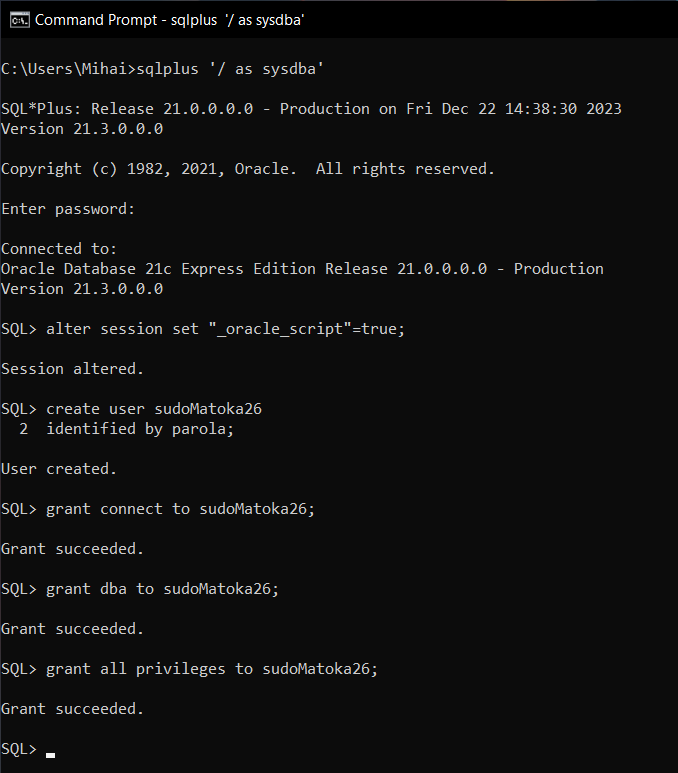
END;

/

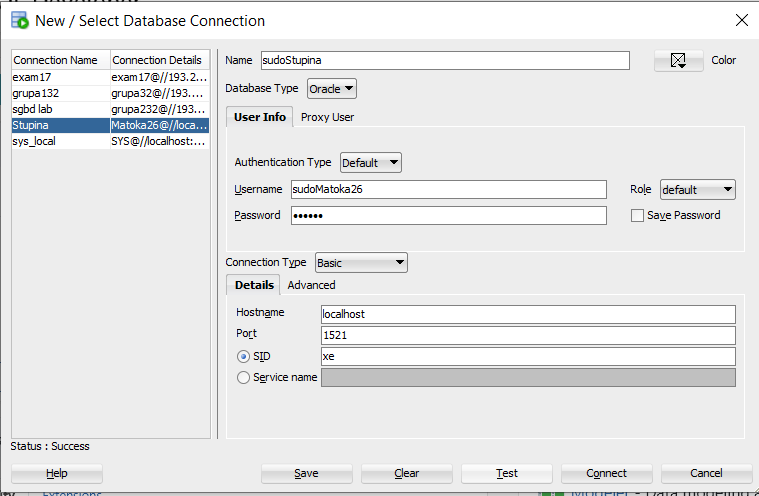
# 12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

Să nu se permită unui utilizator sa folosească comanda DROP decât daca are rolul de DBA(administrator)

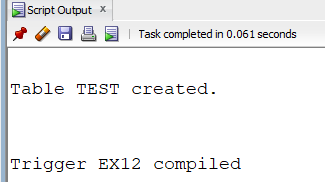
🡪Am creat un utilizator nou și i-am oferit rolul de DBA



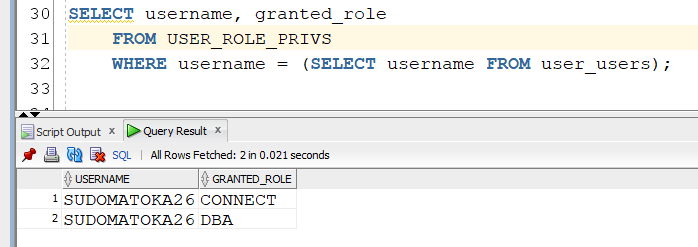
🡪Am creat o nouă conexiune cu acest nou user



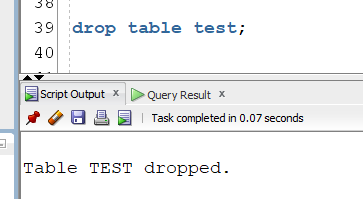
* Am creat o tabelă de test pe care vom aplica operațiile și am compilat trigger-ul



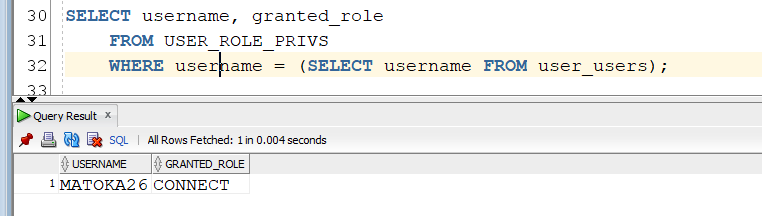
🡪Am verificat rolurile user-ului current, printre care și DBA

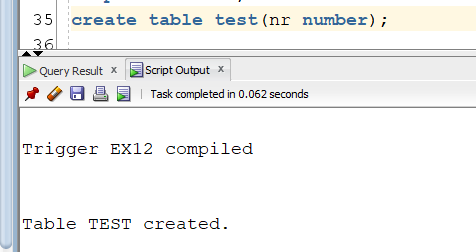


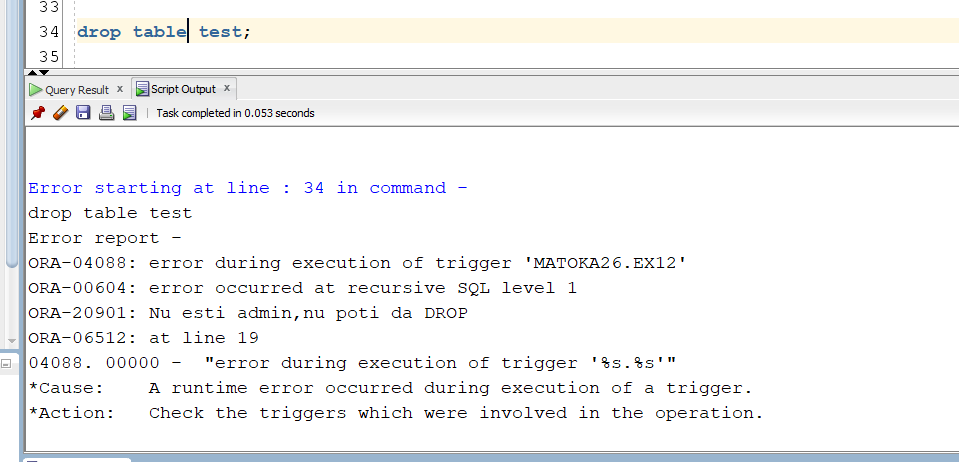
🡪Operația de DROP ne-a fost permisă



🡪Iar acum din ipostaza unui user obișnuit







🡪Nu este permisă comanda DROP

CREATE OR REPLACE TRIGGER ex12

AFTER DROP ON SCHEMA

DECLARE

v\_role user\_role\_privs.granted\_role%TYPE;

BEGIN

--iau rolul userului daca este DBA

SELECT granted\_role

INTO v\_role

FROM USER\_ROLE\_PRIVS

--verific pentru userul curent

WHERE username = (SELECT username FROM user\_users)

AND granted\_role = 'DBA';

--daca nu este DBA ii atribuim o valore

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_role := 'NO\_DBA';

--daca nu este DBA aruncam o eroare

IF v\_role <> 'DBA' THEN

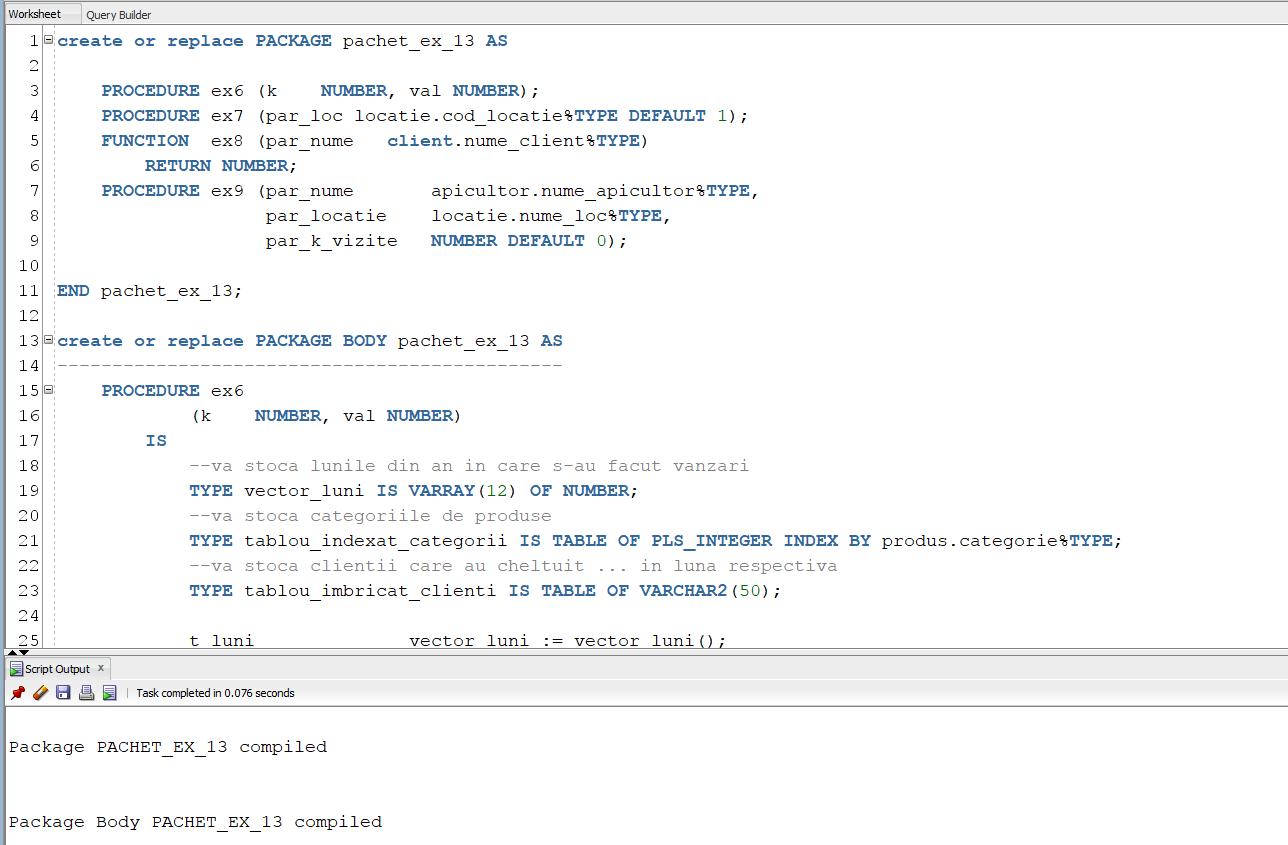
RAISE\_APPLICATION\_ERROR(-20901,'Nu esti admin,nu poti da DROP');

END IF;

END;

/

# 13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.



create or replace PACKAGE pachet\_ex\_13 AS

PROCEDURE ex6 (k NUMBER, val NUMBER);

PROCEDURE ex7 (par\_loc locatie.cod\_locatie%TYPE DEFAULT 1);

FUNCTION ex8 (par\_nume client.nume\_client%TYPE)

RETURN NUMBER;

PROCEDURE ex9 (par\_nume apicultor.nume\_apicultor%TYPE,

par\_locatie locatie.nume\_loc%TYPE,

par\_k\_vizite NUMBER DEFAULT 0);

END pachet\_ex\_13;

/

create or replace PACKAGE BODY pachet\_ex\_13 AS

----------------------------------------------

PROCEDURE ex6

(k NUMBER, val NUMBER)

IS

--va stoca lunile din an in care s-au facut vanzari

TYPE vector\_luni IS VARRAY(12) OF NUMBER;

--va stoca categoriile de produse

TYPE tablou\_indexat\_categorii IS TABLE OF PLS\_INTEGER INDEX BY produs.categorie%TYPE;

--va stoca clientii care au cheltuit ... in luna respectiva

TYPE tablou\_imbricat\_clienti IS TABLE OF VARCHAR2(50);

t\_luni vector\_luni := vector\_luni();

t\_categorii tablou\_indexat\_categorii;

t\_clienti tablou\_imbricat\_clienti;

v\_aux NUMBER;

v\_aux\_nume\_luna VARCHAR2(20);

v\_aux\_nume\_client VARCHAR2(40);

BEGIN

-- ia numarul de vanzari din fiecare luna

FOR i IN 1..k LOOP

--facem un tabel nou pentru fiecare luna

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

t\_luni.extend;

select count(extract(month from data\_vanzare))

into t\_luni(i)

from cumpara

where i = extract(month from data\_vanzare);

--converteste din int in numele lunii

SELECT TO\_CHAR(TO\_DATE(i, 'MM'), 'Month')

into v\_aux\_nume\_luna

from dual;

DBMS\_OUTPUT.PUT\_LINE(v\_aux\_nume\_luna||' vanzari:'||t\_luni(i));

-- cate vanzari au fost din fiecare categorie

FOR v\_categ IN (SELECT DISTINCT nvl(categorie, 'nespecificat') AS category FROM produs) LOOP

SELECT nvl(SUM(c.cantitate),0)

INTO t\_categorii(v\_categ.category)

FROM cumpara c, produs p

WHERE p.cod\_produs = c.cod\_produs

AND nvl(p.categorie, 'nespecificat') = v\_categ.category

AND EXTRACT(MONTH FROM c.data\_vanzare) = i ;

DBMS\_OUTPUT.PUT\_LINE(' '||v\_categ.category||' unitati:'||t\_categorii(v\_categ.category));

END LOOP;

--ia clientii care au facut macar o plata mai mare decat parametru val in acea luna

SELECT DISTINCT cl.nume\_client||' '||cl.prenume\_client

BULK COLLECT INTO t\_clienti

FROM cumpara c, produs p, client cl

WHERE c.cod\_produs = p.cod\_produs

AND cl.id\_client = c.id\_client

AND EXTRACT(MONTH FROM c.data\_vanzare) = i

AND p.pret \* c.cantitate >= val;

IF t\_clienti.COUNT = 0

THEN DBMS\_OUTPUT.PUT\_LINE('Nu a fost niciun client cu o plata asa mare');

ELSE DBMS\_OUTPUT.PUT\_LINE('Clientii care a avut o plata cu suma cel putin'||val||' in luna '||v\_aux\_nume\_luna||':');

FOR i IN 1..t\_clienti.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE(' '||t\_clienti(i));

END LOOP;

END IF;

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

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

END LOOP;

END ex6;

----------------------------------------------

PROCEDURE ex7

(par\_loc locatie.cod\_locatie%TYPE DEFAULT 1)

IS

TYPE refcursor IS REF CURSOR;

CURSOR c (par\_cursor locatie.cod\_locatie%TYPE) IS

-- id media mediei calificativelor

SELECT l.id\_stup ,avg((blandete + instinct\_roire + productie\_miere)/3),

--intoarce produsele recoltate de la stupii respectivi

CURSOR(select p.id\_recoltare,pr.nume\_produs

from familie\_de\_albine f, produce p,produs pr

where f.id\_stup = p.id\_stup

and l.id\_stup = f.id\_stup

and pr.cod\_produs = p.cod\_produs)

FROM verifica l

--verifica locatia potrivita

WHERE par\_cursor = l.cod\_locatie

--verifica anul potrivit

AND EXTRACT(YEAR FROM l.data\_verificare)

= EXTRACT(YEAR FROM sysdate)

GROUP BY l.id\_stup;

v\_cursor refcursor;

v\_id\_stup verifica.id\_stup%TYPE;

v\_medie NUMBER;

v\_id\_rec produce.id\_recoltare%TYPE;

v\_nume\_prod produs.nume\_produs%TYPE;

v\_cnt\_lines NUMBER;

BEGIN

OPEN c(par\_loc);

LOOP

FETCH c INTO v\_id\_stup, v\_medie, v\_cursor;

EXIT WHEN c%NOTFOUND;

v\_cnt\_lines := 0;

DBMS\_OUTPUT.PUT\_LINE('Id stup: '||v\_id\_stup||' Medie califactiv: '||round(v\_medie,2));

LOOP

FETCH v\_cursor INTO v\_id\_rec, v\_nume\_prod;

EXIT WHEN v\_cursor%NOTFOUND;

v\_cnt\_lines := 1;

DBMS\_OUTPUT.PUT\_LINE(' ->Id recoltare: '||v\_id\_rec||' Nume produs: '||v\_nume\_prod);

END LOOP;

--verifica daca cursorul a returnat macar o linie pentru a afisa mesajul

IF v\_cnt\_lines = 0 THEN

DBMS\_OUTPUT.PUT\_LINE(' ->Niciun produs recoltat');

END IF;

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

END LOOP;

CLOSE c;

END ex7;

----------------------------------------------

FUNCTION ex8

(par\_nume client.nume\_client%TYPE)

RETURN NUMBER IS

v\_result NUMBER;

v\_id client.id\_client%TYPE;

BEGIN

-- am folosit clauza de group by pentru a rezolva situatiile conflictuale de nume

SELECT c.id\_client, SUM(c.cantitate \* p.pret)

INTO v\_id,v\_result

FROM cumpara c, client cl, produs p

WHERE cl.id\_client = c.id\_client

AND INITCAP(cl.nume\_client) = INITCAP(par\_nume)

AND c.cod\_produs = p.cod\_produs

group by c.id\_client;

RETURN v\_result;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000,

'Nu exista clientul '||par\_nume);

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001,

'Exista mai multi clineit cu numele '||par\_nume);

WHEN OTHERS THEN

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

END ex8;

----------------------------------------------

PROCEDURE ex9

(par\_nume apicultor.nume\_apicultor%TYPE,

par\_locatie locatie.nume\_loc%TYPE,

par\_k\_vizite NUMBER DEFAULT 0)

IS

v\_id\_apic apicultor.id\_apicultor%TYPE;

v\_cnt\_ang NUMBER;

v\_cnt\_viz NUMBER;

BEGIN

--cati angajati a angajat apicultorul in acea locatie si inca lucreaza

SELECT ap.id\_apicultor,count(a.id\_angajat), count(id\_vizita)

INTO v\_id\_apic, v\_cnt\_ang,v\_cnt\_viz

FROM apicultor ap, locatie l, angajeaza an,angajat a, viziteaza v

WHERE

--cauta apicultor dupa nume

ap.nume\_apicultor = Initcap(par\_nume)

--cauta locatia dupa nume

AND l.nume\_loc = par\_locatie

-- join pentru angajati pe locatie

AND a.id\_angajat = an.id\_angajat

AND a.cod\_locatie = l.cod\_locatie

--join pt count(id\_angajat) ->cati a angajat in acea locatie si inca lucreaza

AND ap.id\_apicultor = an.id\_apicultor

AND an.data\_eliminare is null

--join uri pentru count (id\_vizita)

AND v.id\_apicultor = ap.id\_apicultor

AND v.cod\_locatie = l.cod\_locatie

group by ap.id\_apicultor

having count(id\_vizita) > par\_k\_vizite;

DBMS\_OUTPUT.PUT\_LINE(v\_id\_apic||' '||v\_cnt\_ang);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Niciun apicultor cu acest nume nu corespunde sau locatie invalida');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Mai multi apicultori cu aces nume');

WHEN OTHERS THEN

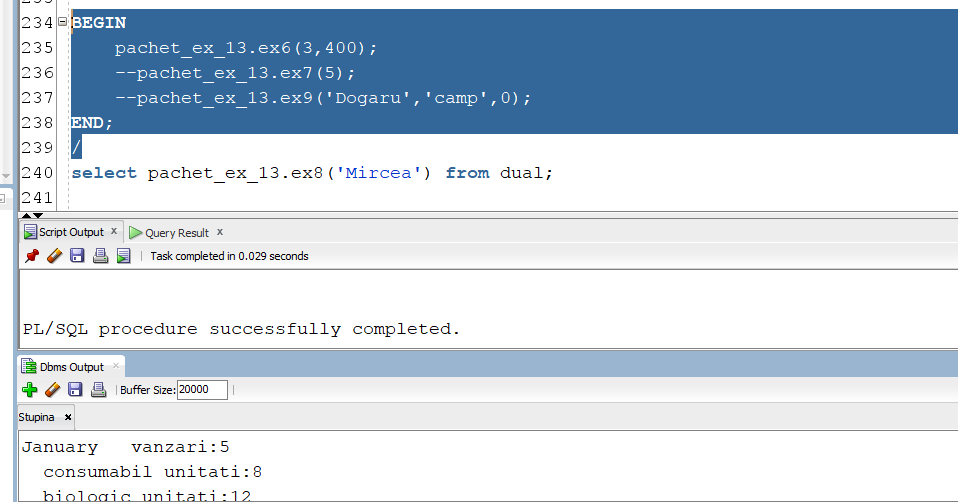
DBMS\_OUTPUT.PUT\_LINE('Alta eroare');

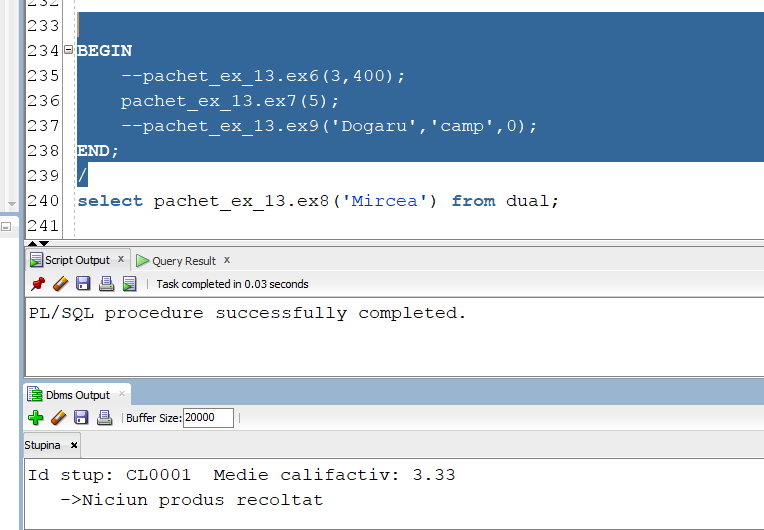
END ex9;

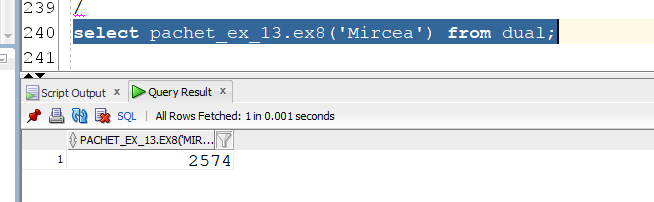
----------------------------------------------

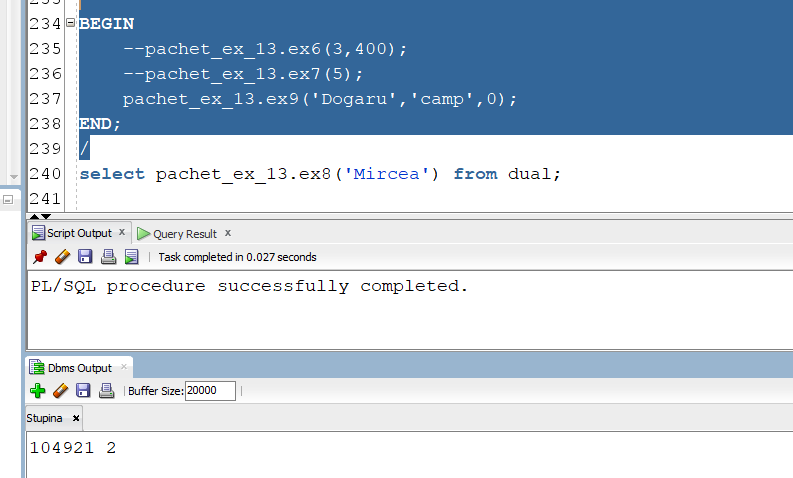
END pachet\_ex\_13;

/





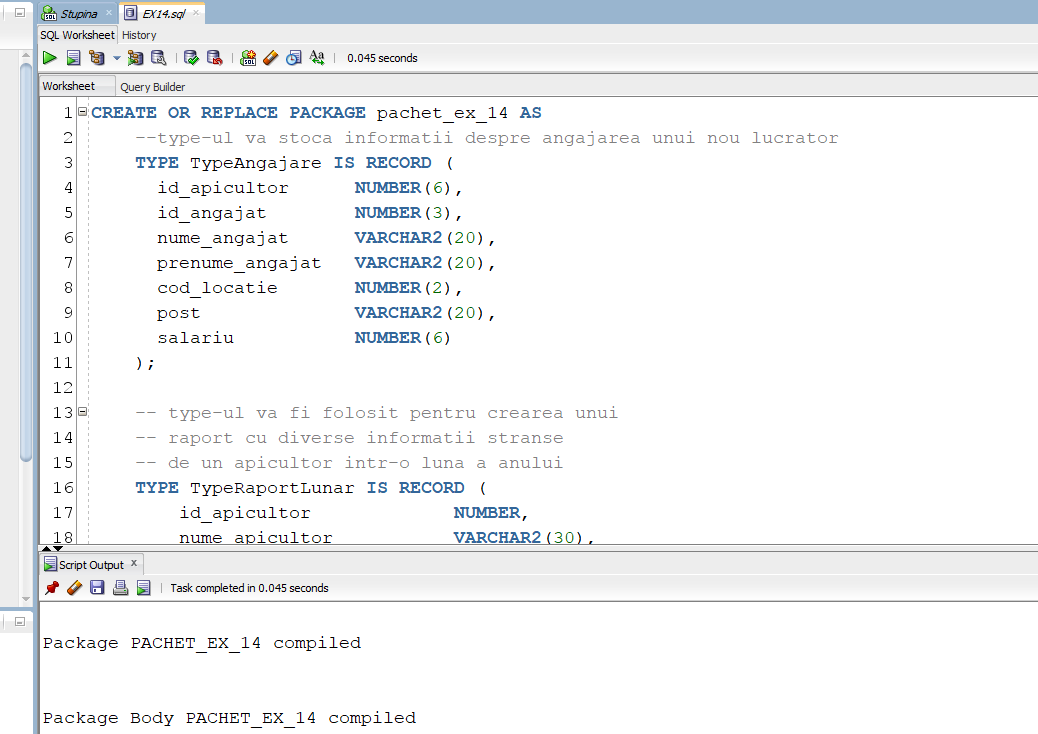




# 14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

Pachetul este destinat unui apicultor pentru a facilita diverse acțiuni specifice acestuia în cadrul stupinei precum:

* Angajarea personalului
* Înlocuirea mătcilor din stupi
* Crearea de raporturi lunare despre activitatea stupinei



CREATE OR REPLACE PACKAGE pachet\_ex\_14 AS

--type-ul va stoca informatii despre angajarea unui nou lucrator

TYPE TypeAngajare IS RECORD (

id\_apicultor NUMBER(6),

id\_angajat NUMBER(3),

nume\_angajat VARCHAR2(20),

prenume\_angajat VARCHAR2(20),

cod\_locatie NUMBER(2),

post VARCHAR2(20),

salariu NUMBER(6)

);

-- type-ul va fi folosit pentru crearea unui

-- raport cu diverse informatii stranse

-- de un apicultor intr-o luna a anului

TYPE TypeRaportLunar IS RECORD (

id\_apicultor NUMBER,

nume\_apicultor VARCHAR2(30),

prenume\_apicultor VARCHAR2(30),

luna\_raport NUMBER,

--an\_raport NUMBER,

numar\_stupi\_verificati NUMBER,

productie\_medie\_miere NUMBER,

comentariu VARCHAR2(255)

);

-- Functia ia un obiect the tip TypeAngajare si returneaza

-- 2 -> daca angajatul nu este inca inregistrat

-- 1 -> daca angajatul a mai fost inregistrat inainte

-- 0 -> daca apare vreo eroare

FUNCTION verificaAngajare (potential\_angajat TypeAngajare)

RETURN NUMBER;

-- Procedura insereaza informatii despre un viitor angajat

-- in tabelele corespunzatoare dupa caz

PROCEDURE angajeazaPersonal

(id\_apicultor apicultor.id\_apicultor%TYPE,

id\_angajat NUMBER,

nume\_angajat VARCHAR2,

prenume\_angajat VARCHAR2,

cod\_locatie NUMBER,

post VARCHAR2,

salariu NUMBER);

-- Procedura va adauga in tabela MATCA o noua matca pentru un stup

-- daca acesta nu avea deja una,

-- daca avea deja una,cea veche va fi marcata ca inlocuita si cea

-- noua va fi inserata in schimb

PROCEDURE schimbaMatca

(p\_id\_stup familie\_de\_albine.id\_stup%TYPE,

p\_id\_matca matca.id\_matca%TYPE);

-- Functia va returna un obiect de tip TypeRaportLunar

-- care va contine informati aferente

FUNCTION IntocmireRaport

(p\_id\_apicultor NUMBER,

p\_data\_raport DATE,

p\_comentariu VARCHAR2)

RETURN TypeRaportLunar;

-- Procedura doar va afisa formatat un raport,

-- primind aceeasi parametrii ca IntocmireRaport(...)

PROCEDURE printRaport

(p\_id\_apicultor NUMBER,

p\_data\_raport DATE,

p\_comentariu VARCHAR2);

-- O varianta overloaded a procedurii de mai sus

-- care ia ca parametru un obiect,dar face exact

-- acelasi lucru

PROCEDURE printRaport

(v\_raport TypeRaportLunar);

END pachet\_ex\_14;

/

CREATE OR REPLACE PACKAGE BODY pachet\_ex\_14 AS

----------------------------------------------

FUNCTION verificaAngajare

(potential\_angajat TypeAngajare)

RETURN NUMBER IS

v\_lucreaza\_in\_prezent DATE;

v\_check\_locatie NUMBER;

v\_check\_apicultor NUMBER;

v\_check\_angajat NUMBER;

CURSOR c (id NUMBER) IS

SELECT data\_eliminare

FROM angajeaza an

WHERE an.id\_angajat = id

ORDER BY data\_angajare DESC;

BEGIN

-- verifica daca apicultorul e valid

Select count(id\_apicultor)

INTO v\_check\_apicultor

from apicultor

where id\_apicultor = potential\_angajat.id\_apicultor;

IF v\_check\_apicultor = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002,'APICULTOR INVALID');

RETURN 0;

END IF;

-- verifica daca locatie e valida

Select count(cod\_locatie)

INTO v\_check\_locatie

from locatie

where cod\_locatie = potential\_angajat.cod\_locatie;

IF v\_check\_locatie = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001,'LOCATIE INVALIDA');

RETURN 0;

END IF;

-- intai verific daca exista deja angajatul intr-o tabela

Select count(id\_angajat)

INTO v\_check\_angajat

from angajat

where id\_angajat = potential\_angajat.id\_angajat;

IF v\_check\_angajat = 0 THEN

RETURN 2;

END IF;

-- datele de eliminare sunt sortate dupa data angajarii

-- ultima data de eliminare va corespune ultimei angajari

-- iar un angajat poate fi angajat o singura data la un

-- moment de timp, deci prima valoare returnata va spune

-- daca lucreaza in prezent(NULL) sau nu(o data)

OPEN c(potential\_angajat.id\_angajat);

FETCH c INTO v\_lucreaza\_in\_prezent;

CLOSE c;

IF v\_lucreaza\_in\_prezent is NULL THEN

RAISE\_APPLICATION\_ERROR(-20000,'ACEST ANGAJAT LUCREAZA DEJA');

RETURN 0;

END IF;

RETURN 1;

END verificaAngajare;

----------------------------------------------

PROCEDURE angajeazaPersonal

(id\_apicultor apicultor.id\_apicultor%TYPE,

id\_angajat NUMBER,

nume\_angajat VARCHAR2,

prenume\_angajat VARCHAR2,

cod\_locatie NUMBER,

post VARCHAR2,

salariu NUMBER)

IS

v\_check NUMBER;

v\_angajat TypeAngajare;

BEGIN

v\_angajat := TypeAngajare(

id\_apicultor => id\_apicultor,

id\_angajat => id\_angajat,

nume\_angajat => nume\_angajat,

prenume\_angajat => prenume\_angajat,

cod\_locatie => cod\_locatie,

post => post,

salariu => salariu

);

-- returneaza 1 daca e ok, else da eroare

SELECT verificaAngajare(v\_angajat)

INTO v\_check

FROM dual;

IF v\_check = 0 THEN

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

-- daca angajatul deja exista trebuie adaugate informatii doar in tabela de angajeaza

ELSE

IF v\_check = 2 THEN

INSERT INTO angajat values(id\_angajat, cod\_locatie,nume\_angajat,prenume\_angajat);

END IF;

INSERT INTO angajeaza values(id\_apicultor, id\_angajat, to\_date(SYSDATE,'dd.mm.yyyy'),NULL,post,salariu);

-- daca nu lucreaza inca deloc, trebuie adaugate info si in tabela de angajat cu numele etc

END IF;

END;

----------------------------------------------

PROCEDURE schimbaMatca

(p\_id\_stup familie\_de\_albine.id\_stup%TYPE,

p\_id\_matca matca.id\_matca%TYPE)

IS

v\_check\_stup NUMBER;

v\_check\_matca NUMBER;

v\_culoare VARCHAR2(1);

BEGIN

SELECT COUNT(p\_id\_Stup)

INTO v\_check\_stup

FROM familie\_de\_albine f

WHERE f.id\_stup = p\_id\_stup;

IF v\_check\_stup = 0 THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Stupul nu exista!');

END IF;

-- verifica daca stupul are deja matca

SELECT COUNT(m.id\_stup)

INTO v\_check\_matca

FROM matca m

WHERE m.id\_stup = p\_id\_Stup;

IF v\_check\_matca > 0 THEN

--daca avea deja matca,marcheaz-o ca inlocuita si insereaz-o pe cea noua

UPDATE matca

SET data\_inlocuire = to\_date(SYSDATE,'dd.mm.yyyy')

WHERE id\_stup = p\_id\_stup and data\_inlocuire is null;

END IF;

--alege culoare conforma anului dupa codul culorilor

SELECT CASE

WHEN MOD(EXTRACT(YEAR FROM SYSDATE),5) = 0 THEN 's'

WHEN MOD(EXTRACT(YEAR FROM SYSDATE),5) = 1 THEN 'a'

WHEN MOD(EXTRACT(YEAR FROM SYSDATE),5) = 2 THEN 'g'

WHEN MOD(EXTRACT(YEAR FROM SYSDATE),5) = 3 THEN 'r'

WHEN MOD(EXTRACT(YEAR FROM SYSDATE),5) = 4 THEN 'v'

END

INTO v\_culoare

FROM DUAL;

INSERT INTO matca VALUES(p\_id\_matca, p\_id\_stup, 'carpatina',v\_culoare,NULL);

END;

----------------------------------------------

FUNCTION IntocmireRaport

(p\_id\_apicultor NUMBER,

p\_data\_raport DATE,

p\_comentariu VARCHAR2)

RETURN TypeRaportLunar

IS

raport TypeRaportLunar := TypeRaportLunar(NULL,NULL,NULL,NULL,0,0,NULL); --ADD 1 parameter later for year

BEGIN

-- completeaza cu info despre apicultor

-- si trateaza exceptia

BEGIN

SELECT id\_apicultor,nume\_apicultor,prenume\_apicultor

INTO raport.id\_apicultor, raport.nume\_apicultor, raport.prenume\_apicultor

FROM apicultor

WHERE id\_apicultor = p\_id\_apicultor;

-- daca id-ul nu este bun

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'APICULTOR INVALID');

END;

-- adauga luna si anul

raport.luna\_raport := EXTRACT(MONTH FROM p\_data\_raport);

--raport.an\_raport := EXTRACT(YEAR FROM p\_data\_raport);

-- adauga nr stupi verificati si media de miere

SELECT count(\*), avg(productie\_miere)

INTO raport.numar\_stupi\_verificati, raport.productie\_medie\_miere

FROM verifica

WHERE id\_apicultor = p\_id\_apicultor

and EXTRACT(MONTH FROM data\_verificare) = raport.luna\_raport;

-- add EXTRACT(YEAR FROM data\_verificare) = raport.an\_raport;

raport.comentariu := p\_comentariu;

RETURN raport;

END IntocmireRaport;

----------------------------------------------

PROCEDURE printRaport

(p\_id\_apicultor NUMBER,

p\_data\_raport DATE,

p\_comentariu VARCHAR2)

IS

v\_raport TypeRaportLunar;

BEGIN

v\_raport := intocmireRaport(p\_id\_apicultor, p\_data\_raport, p\_comentariu);

DBMS\_OUTPUT.PUT\_LINE('ID\_APICULTOR: '||v\_raport.id\_apicultor);

DBMS\_OUTPUT.PUT\_LINE('NUME\_APICULTOR: '||v\_raport.nume\_apicultor);

DBMS\_OUTPUT.PUT\_LINE('PRENUME\_APICULTOR: '||v\_raport.prenume\_apicultor);

DBMS\_OUTPUT.PUT\_LINE('LUNA: '||TO\_CHAR(TO\_DATE(v\_raport.luna\_raport, 'MM'), 'Month'));

--DBMS\_OUTPUT.PUT\_LINE('ANUL: '||TO\_CHAR(TO\_DATE(v\_raport.an\_raport, 'YY'), 'Year'));

DBMS\_OUTPUT.PUT\_LINE('NUMAR STUPI VERIFICATI: '||v\_raport.numar\_stupi\_verificati);

DBMS\_OUTPUT.PUT\_LINE('MEDIE PRODUCTIE MIERE: '||v\_raport.productie\_medie\_miere);

DBMS\_OUTPUT.PUT\_LINE('OBSERVATII: '||v\_raport.comentariu);

END;

----------------------------------------------

PROCEDURE printRaport

(v\_raport TypeRaportLunar)

IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('ID\_APICULTOR: '||v\_raport.id\_apicultor);

DBMS\_OUTPUT.PUT\_LINE('NUME\_APICULTOR: '||v\_raport.nume\_apicultor);

DBMS\_OUTPUT.PUT\_LINE('PRENUME\_APICULTOR: '||v\_raport.prenume\_apicultor);

DBMS\_OUTPUT.PUT\_LINE('LUNA: '||TO\_CHAR(TO\_DATE(v\_raport.luna\_raport, 'MM'), 'Month'));

--DBMS\_OUTPUT.PUT\_LINE('ANUL: '||TO\_CHAR(TO\_DATE(v\_raport.an\_raport, 'YY'), 'Year'));

DBMS\_OUTPUT.PUT\_LINE('NUMAR STUPI VERIFICATI: '||v\_raport.numar\_stupi\_verificati);

DBMS\_OUTPUT.PUT\_LINE('MEDIE PRODUCTIE MIERE: '||v\_raport.productie\_medie\_miere);

DBMS\_OUTPUT.PUT\_LINE('OBSERVATII: '||v\_raport.comentariu);

END;

----------------------------------------------

END pachet\_ex\_14;

/

