Feta Almar-Eran

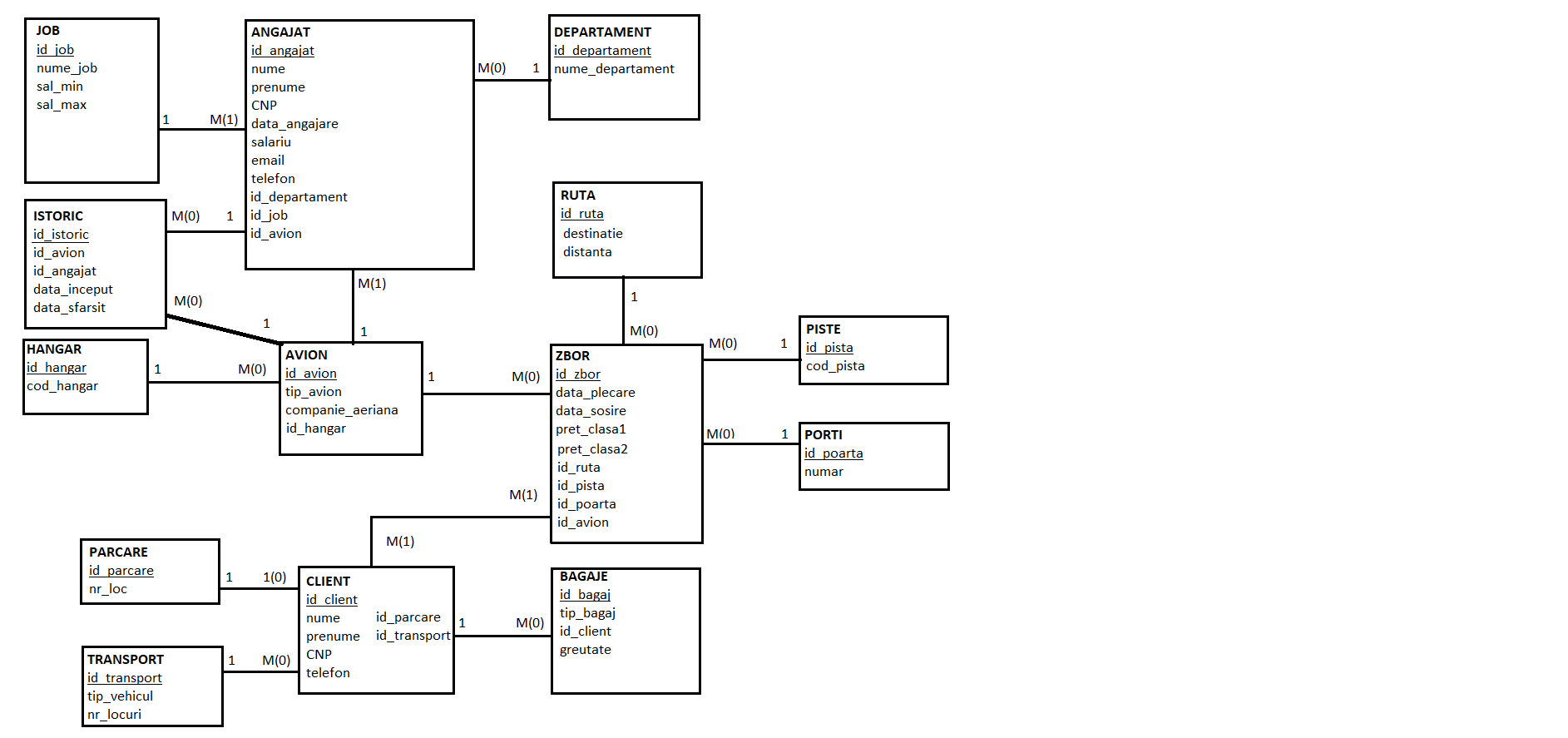
Grupa 233

**Proiect SGBD – Gestionarea unui aeroport**

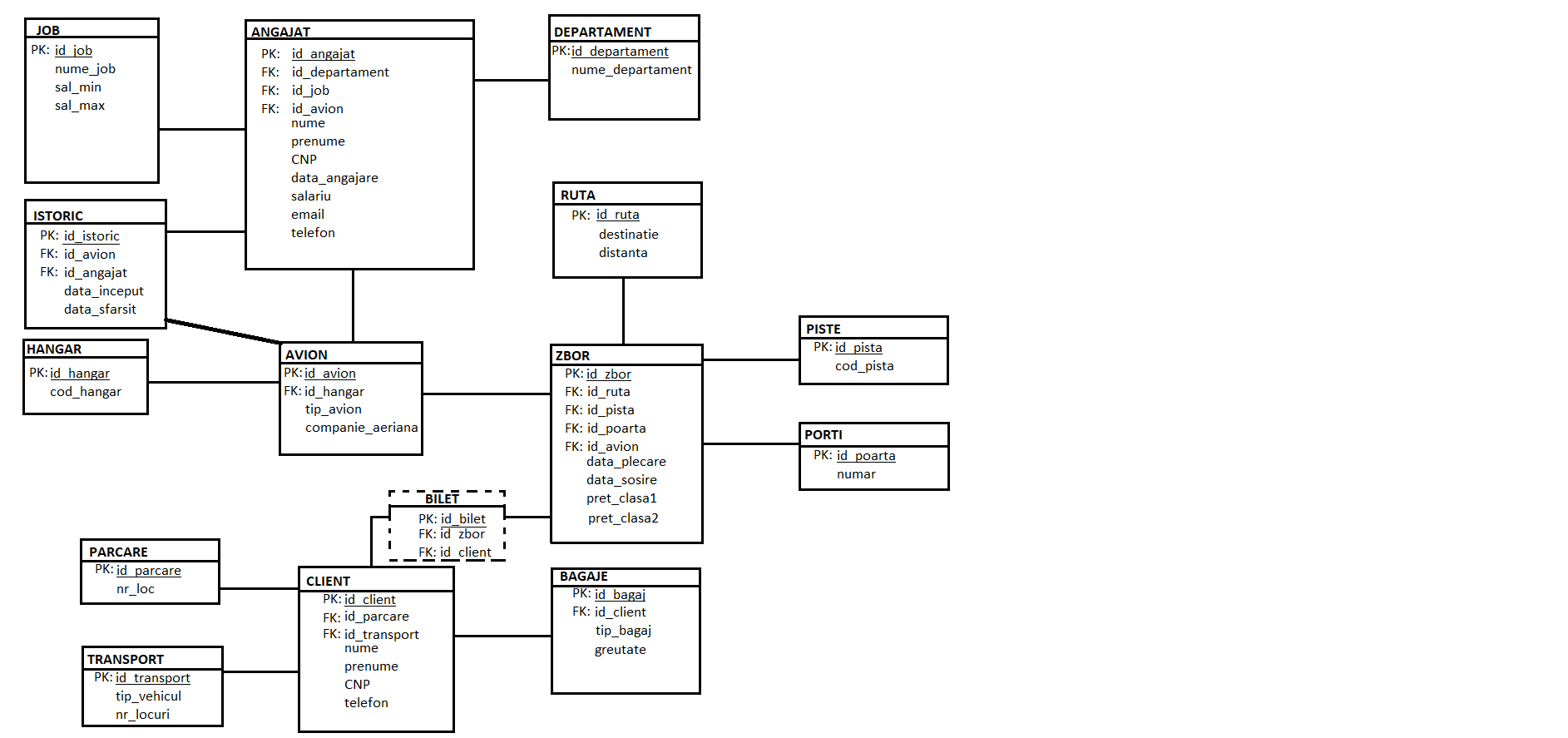
1. **Prezentați pe scurt baza de date (utilitatea ei).**

Mi-am propus sa realizez o baza de date care sa faciliteze gestiunea unui aeroport. Acesta va contine date despre angajatii din aeroport si joburile lor, atat cei care lucreaza in departamente terestre, cat si cei care zboara (pentru acestia se poate tine si evidenta avioanelor in care a mai lucrat pana atunci). De asemenea, baza de date va oferi informatii despre zborurile care pleaca din acel aeroport, avioanele implicate in zboruri, rutele si pistele pe care vor pleca, precum si hangarele si portile unde se afla. Se pot furniza informatii si despre clientii care vor pleca din acel aeroport, alturi de optiunile lor: pot cere sau nu transport pana la aeroport, parcare pentru timp indelungat, sau pot avea bagaje de diferite dimensiuni. Consider ca o astfel de baza de date este utila unui aeroport, intrucat exista foarte multe date care trebuiesc procesate cat mai rapid pentru a facilita fluiditatea activitatilor desfasurate.

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 JOB (

id\_job number(4) constraint pk\_job primary key,

nume\_job varchar2(20) constraint null\_nume\_job not null,

sal\_min number(8,2),

sal\_max number(8,2)

);

create table DEPARTAMENT (

id\_departament number(4) constraint pk\_departament primary key,

nume\_departament varchar2(20) constraint null\_nume\_departament not null

);

create table HANGAR (

id\_hangar number(4) constraint pk\_hangar primary key,

cod\_hangar varchar2(10) constraint null\_cod\_hangar not null

);

create table RUTA (

id\_ruta number(4) constraint pk\_ruta primary key,

destinatie varchar2(20) constraint null\_destinatie not null,

distanta number(6,2)

);

create table PISTE (

id\_pista number(4) constraint pk\_pista primary key,

cod\_pista varchar2(10) constraint null\_cod\_pista not null

);

create table PORTI (

id\_poarta number(4) constraint pk\_poarta primary key,

numar varchar2(10) constraint null\_numar not null

);

create table PARCARE (

id\_parcare number(4) constraint pk\_parcare primary key,

numar\_loc varchar2(10) constraint null\_numar\_loc not null

);

create table TRANSPORT (

id\_transport number(4) constraint pk\_transport primary key,

tip\_vehicul varchar2(10) constraint null\_tip\_vehicul not null,

nr\_locuri number(2)

);

create table AVION (

id\_avion varchar2(10) constraint pk\_avion primary key,

tip\_avion varchar2(20) constraint null\_tip\_avion not null,

companie\_aeriana varchar2(20) constraint null\_companie\_aeriana not null,

id\_hangar number(4),

constraint fk\_id\_hangar foreign key(id\_hangar) references HANGAR(id\_hangar)

);

create table ANGAJAT (

id\_angajat number(4) constraint pk\_angajat primary key,

nume varchar2(20) constraint null\_nume not null,

prenume varchar2(20) constraint null\_prenume not null,

CNP number(13) constraint null\_cnp not null,

data\_angajare date default sysdate constraint null\_data\_angajare not null,

salariu number(8,2),

email varchar2(30),

telefon number(15),

id\_departament number(4),

id\_job number(4) constraint null\_id\_job not null,

id\_avion varchar2(10),

constraint fk\_id\_departament foreign key(id\_departament) references DEPARTAMENT(id\_departament),

constraint fk\_id\_job foreign key(id\_job) references JOB(id\_job),

constraint fk\_id\_avion foreign key(id\_avion) references AVION(id\_avion)

);

create table ISTORIC (

id\_istoric number(4) constraint pk\_istoric primary key,

id\_angajat number(4) constraint null\_id\_angajat not null,

id\_avion varchar2(10) constraint null\_id\_avion not null,

data\_inceput date default sysdate,

data\_sfarsit date default sysdate,

constraint fk\_id\_angajat foreign key(id\_angajat) references ANGAJAT(id\_angajat),

constraint fk\_id\_avion\_istoric foreign key(id\_avion) references AVION(id\_avion)

);

create table ZBOR (

id\_zbor varchar2(10) constraint pk\_zbor primary key,

data\_plecare date default sysdate constraint null\_data\_plecare not null,

data\_sosire date default sysdate constraint null\_date\_sosire not null,

pret\_clasa1 number(6,2),

pret\_clasa2 number(6,2),

id\_avion varchar2(10) constraint null\_id\_avion\_zbor not null,

id\_ruta number(4) constraint null\_id\_ruta not null,

id\_pista number(4) constraint null\_id\_pista not null,

id\_poarta number(4) constraint null\_id\_poarta not null,

constraint fk\_id\_avion\_zbor foreign key(id\_avion) references AVION(id\_avion),

constraint fk\_id\_ruta foreign key(id\_ruta) references RUTA(id\_ruta),

constraint fk\_id\_pista foreign key(id\_pista) references PISTE(id\_pista),

constraint fk\_id\_poarta foreign key(id\_poarta) references PORTI(id\_poarta)

);

create table CLIENT (

id\_client number(4) constraint pk\_client primary key,

nume varchar(20) constraint null\_nume\_client not null,

prenume varchar(20) constraint null\_prenume\_client not null,

CNP number(13) constraint null\_cnp\_client not null,

telefon number(15),

id\_zbor varchar2(10) constraint null\_id\_zbor not null,

id\_parcare number(4),

id\_transport number(4),

constraint fk\_id\_zbor foreign key(id\_zbor) references ZBOR(id\_zbor),

constraint fk\_id\_parcare foreign key(id\_parcare) references PARCARE(id\_parcare),

constraint fk\_id\_transport foreign key(id\_transport) references TRANSPORT(id\_transport)

);

create table BAGAJE (

id\_bagaj number(4) constraint pk\_bagajt primary key,

tip\_bagaj varchar2(10) constraint null\_tip\_bagaj not null,

id\_client number(4) constraint null\_id\_client not null,

greutate number(4,2),

constraint fk\_id\_clinet foreign key(id\_client) references CLIENT(id\_client)

);

create table BILET (

id\_bilet number(4) constraint pk\_bilet primary key,

id\_zbor varchar2(10) constraint null\_zbor\_bilet not null,

id\_client number(4) constraint null\_client\_bilet not null,

constraint fk\_id\_zbor\_bilet foreign key(id\_zbor) references ZBOR(id\_zbor),

constraint fk\_id\_client\_bilet foreign key(id\_client) references CLIENT(id\_client)

);

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

insert into JOB

values (101, 'Pilot', 11000, 15000);

insert into JOB

values (102, 'Insotitor de zbor', 3000, 8000);

insert into JOB

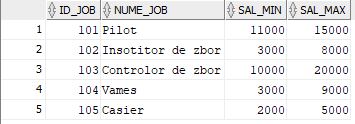
values (103, 'Controlor de zbor', 10000, 20000);

insert into JOB

values (104, 'Vames', 3000, 9000);

insert into JOB

values (105, 'Casier', 2000, 5000);



insert into DEPARTAMENT

values (201, 'Deplasari');

insert into DEPARTAMENT

values (202, 'Control trafic si IT');

insert into DEPARTAMENT

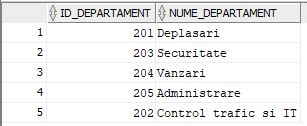
values (203, 'Securitate');

insert into DEPARTAMENT

values (204, 'Vanzari');

insert into DEPARTAMENT

values (205, 'Administrare');



insert into HANGAR

values (301, 'H100');

insert into HANGAR

values (302, 'H230');

insert into HANGAR

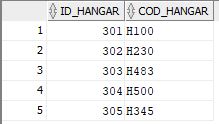
values (303, 'H483');

insert into HANGAR

values (304, 'H500');

insert into HANGAR

values (305, 'H345');



insert into AVION

values ('B\_101', 'Boeing', 'WizzAir', 301);

insert into AVION

values ('B\_102', 'Boeing', 'BlueAir', 303);

insert into AVION

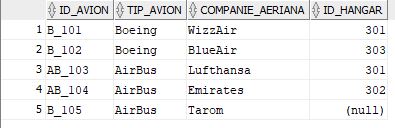
values ('AB\_103', 'AirBus', 'Lufthansa', 301);

insert into AVION

values ('AB\_104', 'AirBus', 'Emirates', 302);

insert into AVION

values ('B\_105', 'AirBus', 'Tarom', null);



insert into ANGAJAT

values (401, 'Barbu', 'Sebastian', 5010101132843, to\_date('12.02.2005','dd-mm-yyyy'), 11000, null, 40767823456, 201, 101, 'B\_102');

insert into ANGAJAT

values (402, 'Ciuraru', 'Mihai', 5010202132854, to\_date('02.12.2006','dd-mm-yyyy'), 3000, 'ciurarumihai@gmail.com', null, 203, 104, null);

insert into ANGAJAT

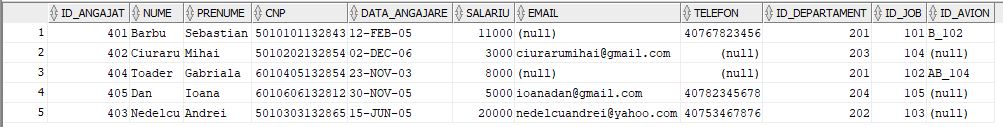
values (403, 'Nedelcu', 'Andrei', 5010303132865, to\_date('15.06.2005','dd-mm-yyyy'), 20000, 'nedelcuandrei@yahoo.com', 40753467876, 202, 103, null);

insert into ANGAJAT

values (404, 'Toader', 'Gabriala', 6010405132854, to\_date('23.11.2003','dd-mm-yyyy'), 8000, null, null, 201, 102, 'AB\_104');

insert into ANGAJAT

values (405, 'Dan', 'Ioana', 6010606132812, to\_date('30.11.2005', 'dd-mm-yyyy'), 5000, 'ioanadan@gmail.com', 40782345678, 204, 105, null);



insert into ISTORIC

values (1, 401, 'B\_101', to\_date('12.02.2005', 'dd-mm-yyyy'), to\_date('13.04.2006', 'dd-mm-yyyy'));

insert into ISTORIC

values (2, 401, 'AB\_103', to\_date('13.04.2006', 'dd-mm-yyyy'), to\_date('20.12.2008', 'dd-mm-yyyy'));

insert into ISTORIC

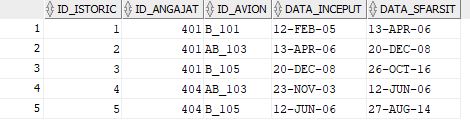
values (3, 401, 'B\_105', to\_date('20.12.2008', 'dd-mm-yyyy'), to\_date('26.10.2016', 'dd-mm-yyyy'));

insert into ISTORIC

values (4, 404, 'AB\_103', to\_date('23.11.2003', 'dd-mm-yyyy'), to\_date('12.06.2006', 'dd-mm-yyyy'));

insert into ISTORIC

values (5, 404, 'B\_105', to\_date('12.06.2006', 'dd-mm-yyyy'), to\_date('27.08.2014', 'dd-mm-yyyy'));



insert into RUTA

values (501, 'Roma', 1500);

insert into RUTA

values (502, 'Paris', 2000);

insert into RUTA

values (503, 'Moscova', 3000);

insert into RUTA

values (504, 'Madrid', 2500);

insert into RUTA

values (505, 'Dubai', 5000);



insert into PISTE

values (601, 'P\_433');

insert into PISTE

values (602, 'P\_656');

insert into PISTE

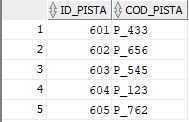
values (603, 'P\_545');

insert into PISTE

values (604, 'P\_123');

insert into PISTE

values (605, 'P\_762');



insert into PORTI

values (701, 'A');

insert into PORTI

values (702, 'B');

insert into PORTI

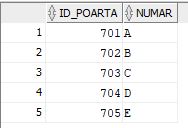
values (703, 'C');

insert into PORTI

values (704, 'D');

insert into PORTI

values (705, 'E');



insert into ZBOR

values ('Z\_101', to\_date('12.02.2021', 'dd-mm-yyyy'), to\_date('12.02.2021', 'dd-mm-yyyy'), 200, 100, 'B\_101', 501, 601, 701);

insert into ZBOR

values ('Z\_102', to\_date('02.04.2021', 'dd-mm-yyyy'), to\_date('03.04.2021', 'dd-mm-yyyy'), 250, 150, 'B\_102', 502, 604, 702);

insert into ZBOR

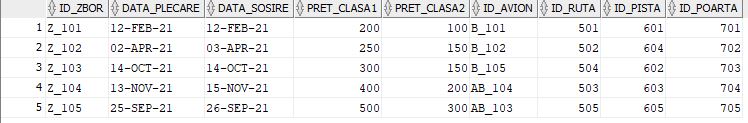
values ('Z\_103', to\_date('14.10.2021', 'dd-mm-yyyy'), to\_date('14.10.2021', 'dd-mm-yyyy'), 300, 150, 'B\_105', 504, 602, 703);

insert into ZBOR

values ('Z\_104', to\_date('13.11.2021', 'dd-mm-yyyy'), to\_date('15.11.2021', 'dd-mm-yyyy'), 400, 200, 'AB\_104', 503, 603, 704);

insert into ZBOR

values ('Z\_105', to\_date('25.09.2021', 'dd-mm-yyyy'), to\_date('26.09.2021', 'dd-mm-yyyy'), 500, 300, 'AB\_103', 505, 605, 705);



insert into PARCARE

values (801, 'A101');

insert into PARCARE

values (802, 'B201');

insert into PARCARE

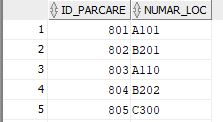
values (803, 'A110');

insert into PARCARE

values (804, 'B202');

insert into PARCARE

values (805, 'C300');



insert into TRANSPORT

values (901, 'Autobuz', 50);

insert into TRANSPORT

values (902, 'Autobuz', 30);

insert into TRANSPORT

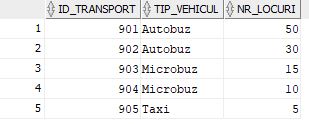
values (903, 'Microbuz', 15);

insert into TRANSPORT

values (904, 'Microbuz', 10);

insert into TRANSPORT

values (905, 'Taxi', 5);



insert into CLIENT

values (1001, 'Ion', 'George', 5010203040506, 40712345678, null, 801, 905);

insert into CLIENT

values (1002, 'Vrinceanu', 'Costin', 5010203123456, 40712365478, null, null, 901);

insert into CLIENT

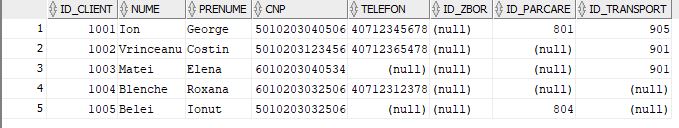
values (1003, 'Matei', 'Elena', 6010203040534, null, null, null, 901);

insert into CLIENT

values (1004, 'Blenche', 'Roxana', 6010203032506, 40712312378, null, null, null);

insert into CLIENT

values (1005, 'Belei', 'Ionut', 5010203032506, null, null, 804, null);



insert into BAGAJE

values(1101, 'De cala', 1001, 30);

insert into BAGAJE

values(1102, 'De cala', 1002, 24);

insert into BAGAJE

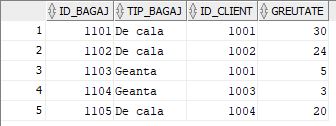
values(1103, 'Geanta', 1001, 5);

insert into BAGAJE

values(1104, 'Geanta', 1003, 3);

insert into BAGAJE

values(1105, 'De cala', 1004, 20);



insert into BILET

values (1201, 'Z\_101', 1001);

insert into BILET

values (1202, 'Z\_101', 1002);

insert into BILET

values (1203, 'Z\_102', 1003);

insert into BILET

values (1204, 'Z\_102', 1004);

insert into BILET

values (1205, 'Z\_103', 1005);

insert into BILET

values (1206, 'Z\_103', 1001);

insert into BILET

values (1207, 'Z\_104', 1002);

insert into BILET

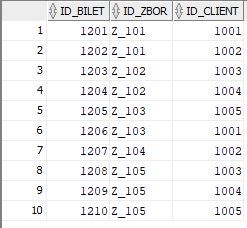
values (1208, 'Z\_105', 1003);

insert into BILET

values (1209, 'Z\_105', 1004);

insert into BILET

values (1210, 'Z\_105', 1005);



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

Creati un subprogram stocat care sa retina in doua tipuri diferite de colectii salariul angajatilor care efectueaza zboruri si cei care nu. Afisati cu ajutorul colectiilor, salariul lor aplicand o majorare cu 10% celor care zboara si cu 5% celor care nu.

CREATE OR REPLACE PROCEDURE ex6

IS

TYPE tablou\_indexat IS TABLE OF angajat.salariu%TYPE INDEX BY PLS\_INTEGER;

t1 tablou\_indexat;

TYPE tablou\_imbricat IS TABLE OF angajat.salariu%TYPE;

t2 tablou\_imbricat := tablou\_imbricat();

BEGIN

SELECT salariu

BULK COLLECT INTO t1

FROM angajat

WHERE id\_avion IS NULL;

SELECT salariu

BULK COLLECT INTO t2

FROM angajat

WHERE id\_avion IS NOT NULL;

DBMS\_OUTPUT.PUT\_LINE('Salariul nou al angajatilor care nu efectueaza zboruri:');

FOR i IN t1.first .. t1.last LOOP

DBMS\_OUTPUT.PUT\_LINE(0.05\*t1(i) + t1(i));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Salariul nou al angajatilor care efectueaza zboruri:');

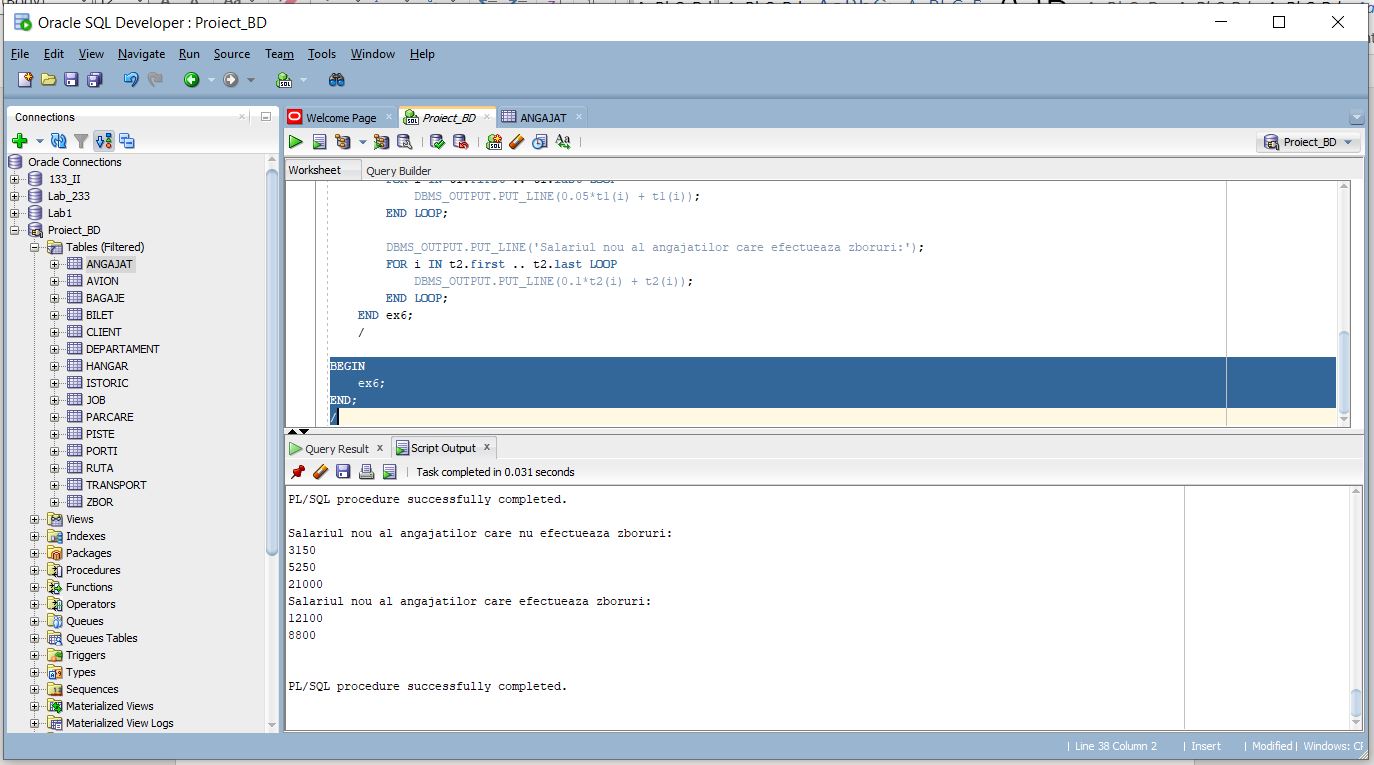
FOR i IN t2.first .. t2.last LOOP

DBMS\_OUTPUT.PUT\_LINE(0.1\*t2(i) + t2(i));

END LOOP;

END ex6;

/



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.**

Folosind o procedura stocata, afisati toate id-urile zborurilor disponibile pentru fiecare ruta existenta, cu ajutorul unui cursor.

CREATE OR REPLACE PROCEDURE ex7

IS

dest\_prec ruta.destinatie%TYPE := '0';

CURSOR c IS

SELECT ruta.destinatie NUME\_RUTA, zbor.id\_zbor ZBOR\_ID

FROM ruta, zbor

WHERE ruta.id\_ruta = zbor.id\_ruta

ORDER BY NUME\_RUTA;

BEGIN

FOR i IN c LOOP

IF dest\_prec = '0' OR dest\_prec != i.NUME\_RUTA THEN

DBMS\_OUTPUT.PUT\_LINE('Zboruri catre ' || i.NUME\_RUTA || ':');

DBMS\_OUTPUT.PUT\_LINE(i.ZBOR\_ID);

dest\_prec := i.NUME\_RUTA;

ELSE

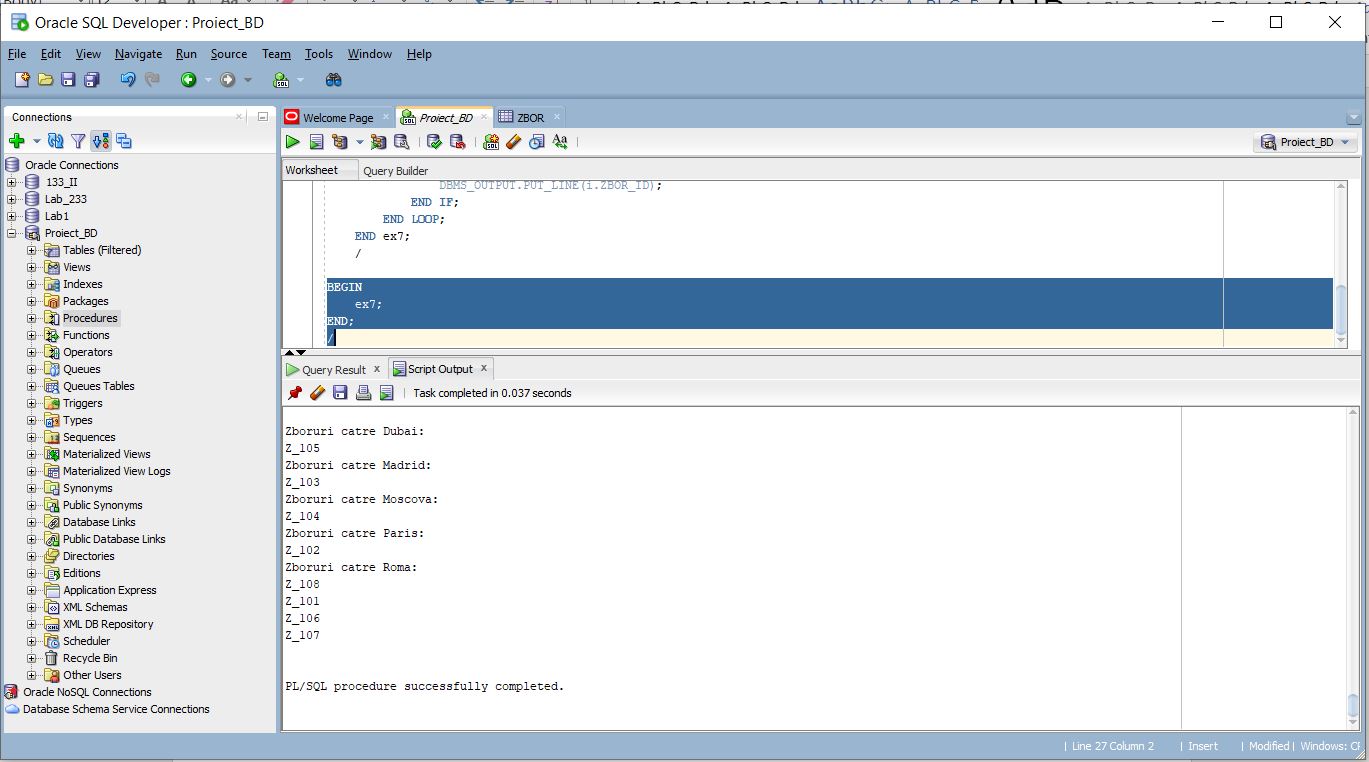
DBMS\_OUTPUT.PUT\_LINE(i.ZBOR\_ID);

END IF;

END LOOP;

END ex7;

/



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

Folositi o functie stocata pentru a returna si afisa compania aeriana a avionului care pleaca catre o destinatie data ca parametru. Afisati eroare in cazul in care exista mai multe companii aeriene care efectueaza un zbor pe acea ruta sau daca nu exista ruta data.

CREATE OR REPLACE FUNCTION ex8

(dest ruta.destinatie%TYPE)

RETURN avion.companie\_aeriana%TYPE

IS

comp avion.companie\_aeriana%TYPE;

BEGIN

SELECT avion.companie\_aeriana

INTO comp

FROM avion, zbor, ruta

WHERE avion.id\_avion = zbor.id\_avion AND

zbor.id\_ruta = ruta.id\_ruta AND

ruta.destinatie = dest;

RETURN comp;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu sunt zboruri catre ruta selectata');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe zboruri catre ruta selectata');

WHEN OTHERS THEN

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

END ex8;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ex8('Roma')); --mai multe zboruri

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ex8('Dubai')); --afiseaza corect

END;

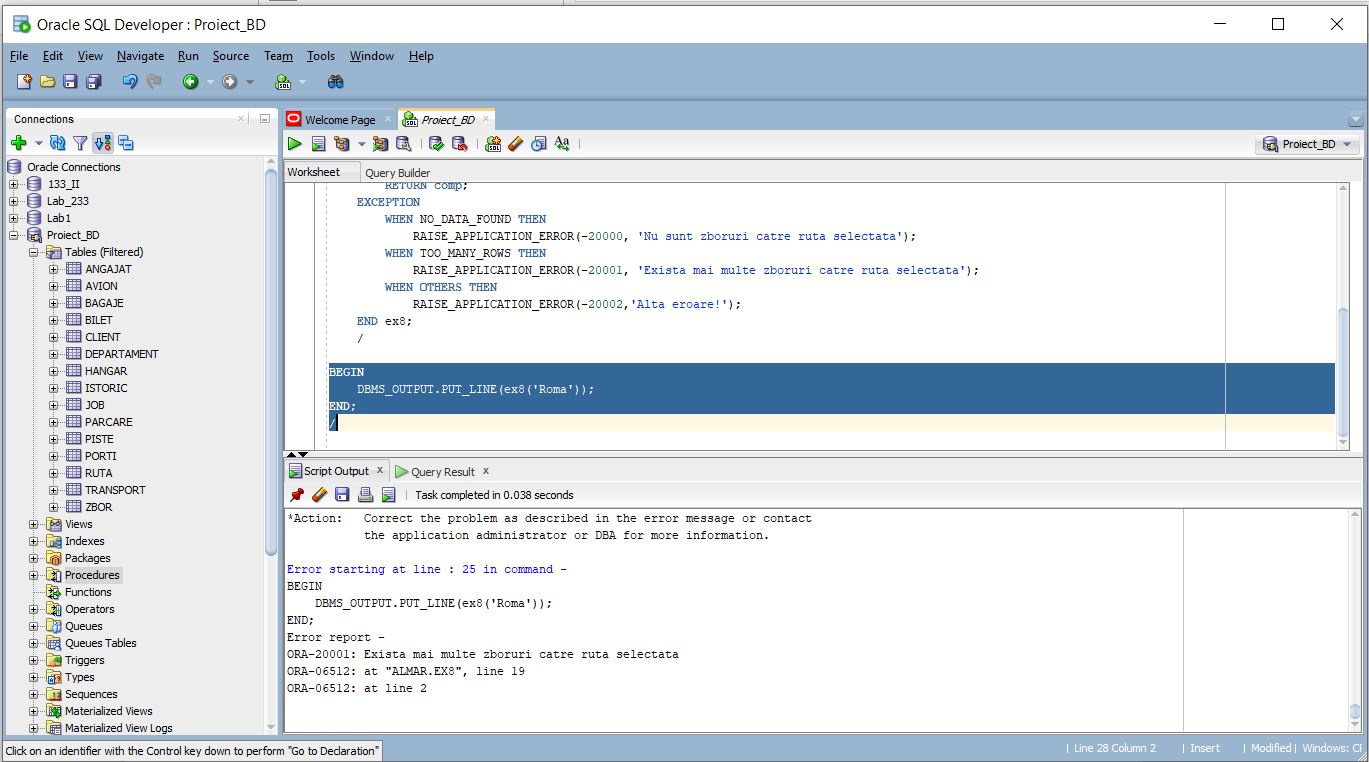
/

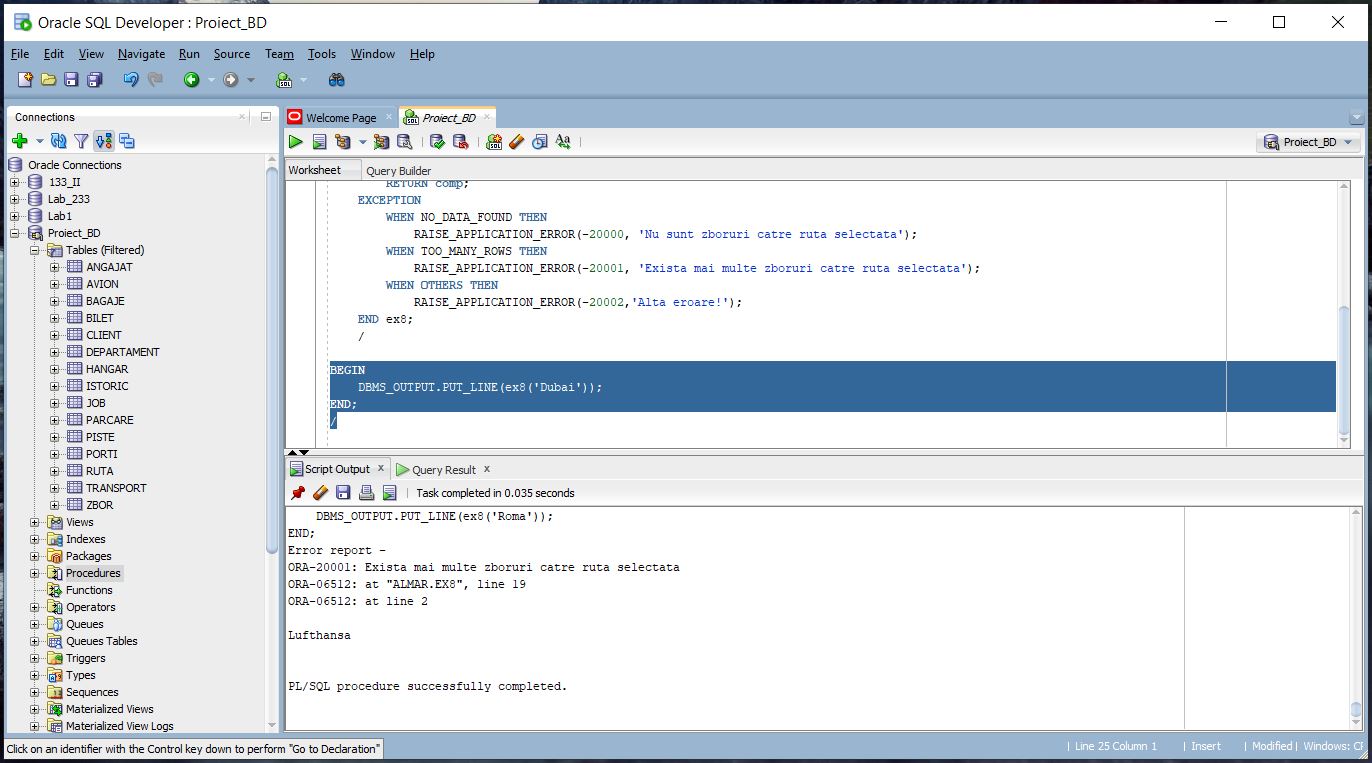
BEGIN

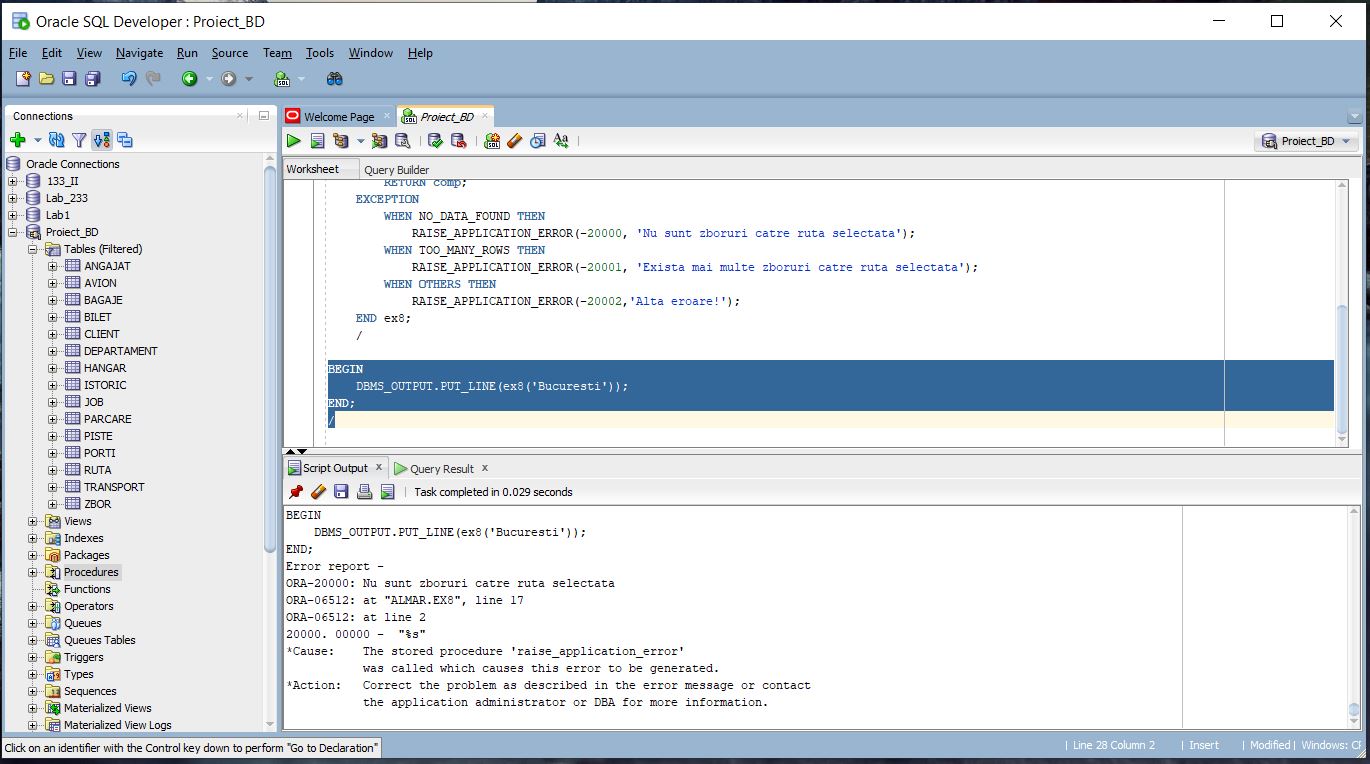
DBMS\_OUTPUT.PUT\_LINE(ex8('Bucuresti')); --nu exista zbor

END;

/







1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat 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.**

Folositi o procedura stocata care primeste ca parametru numele unui client si afiseaza id-ul zborului sau, data de plecare, destinatia si compania aeriana cu care va zbura. Afisati o eroare in cazul in care clientul are mai multe bilete cumparate si daca nu exista clientul dat in baza de date.

CREATE OR REPLACE PROCEDURE ex9

(cl client.nume%TYPE)

IS

id\_z bilet.id\_zbor%TYPE;

data\_plec zbor.data\_plecare%TYPE;

dest ruta.destinatie%TYPE;

comp avion.companie\_aeriana%TYPE;

BEGIN

SELECT bilet.id\_zbor, zbor.data\_plecare, ruta.destinatie, avion.companie\_aeriana

INTO id\_z, data\_plec, dest, comp

FROM client, bilet, zbor, ruta, avion

WHERE client.id\_client = bilet.id\_client and

bilet.id\_zbor = zbor.id\_zbor and

zbor.id\_ruta = ruta.id\_ruta and

zbor.id\_avion = avion.id\_avion and

client.nume = cl;

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

DBMS\_OUTPUT.PUT\_LINE('ID zbor: ' || id\_z);

DBMS\_OUTPUT.PUT\_LINE('Data de plecare: ' || data\_plec);

DBMS\_OUTPUT.PUT\_LINE('Destinatia: ' || dest);

DBMS\_OUTPUT.PUT\_LINE('Compania aeriana: ' || comp);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista client cu acest nume');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul are mai mult de un bilet');

WHEN OTHERS THEN

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

END ex9;

/

BEGIN

ex9('Vrinceanu'); --are mai multe bilete

END;

/

BEGIN

ex9('Ion'); --afiseaza corect

END;

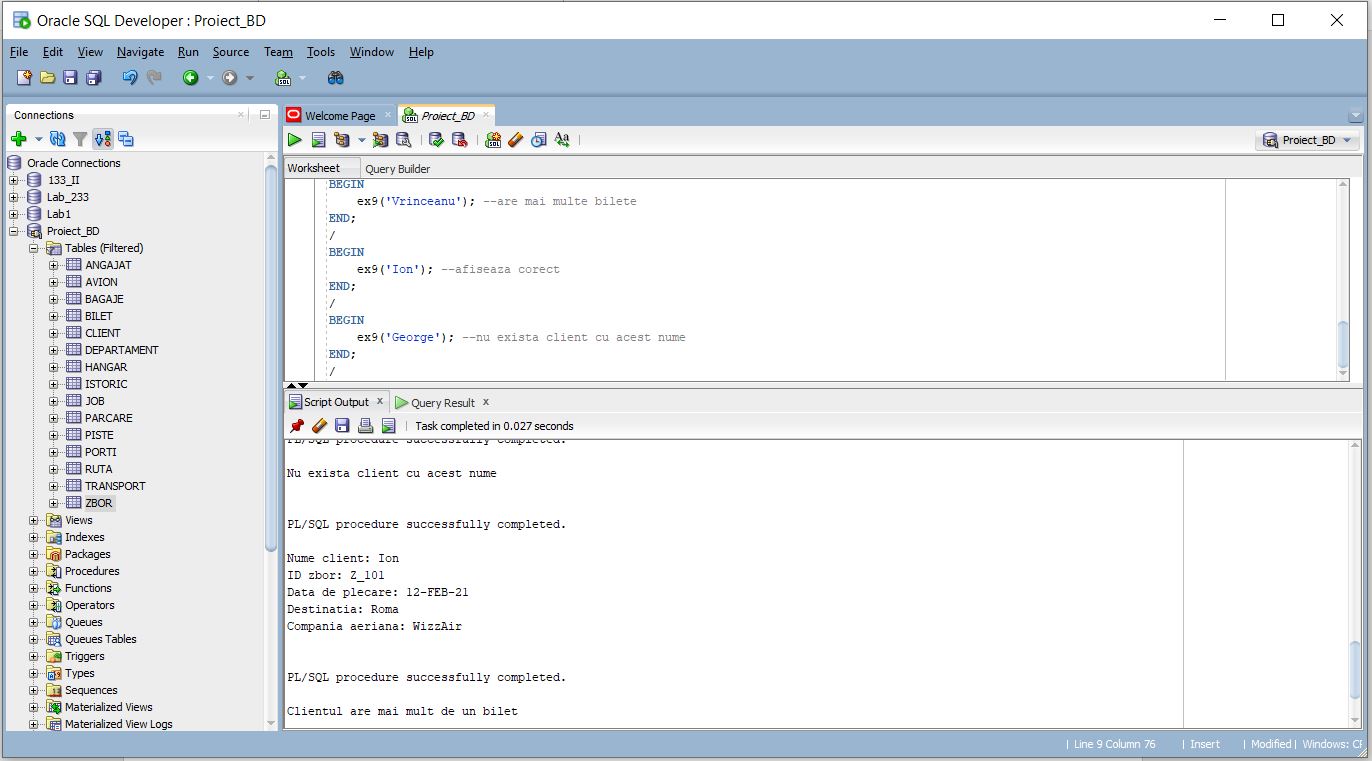
/

BEGIN

ex9('George'); --nu exista client cu acest nume

END;

/



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

Definiti un trigger de tip LMD care sa nu permita inserarea, stergerea sau modificarea datelor din tabela angajati in afara orelor de lucru sau in timpul zilelor nelucratoare.

CREATE OR REPLACE TRIGGER ex10

BEFORE INSERT OR DELETE OR UPDATE ON angajat

BEGIN

IF (TO\_CHAR(sysdate, 'D') = 1) OR

(TO\_CHAR(sysdate, 'DD/MM') = '01/01') OR

(TO\_CHAR(sysdate, 'DD/MM') = '01/05') OR

(TO\_CHAR(sysdate, 'DD/MM') = '01/06') OR

(TO\_CHAR(sysdate, 'DD/MM') = '01/12') OR

(TO\_CHAR(sysdate, 'DD/MM') = '25/12') THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Baza de date nu poate fi actualizata in zilele libere');

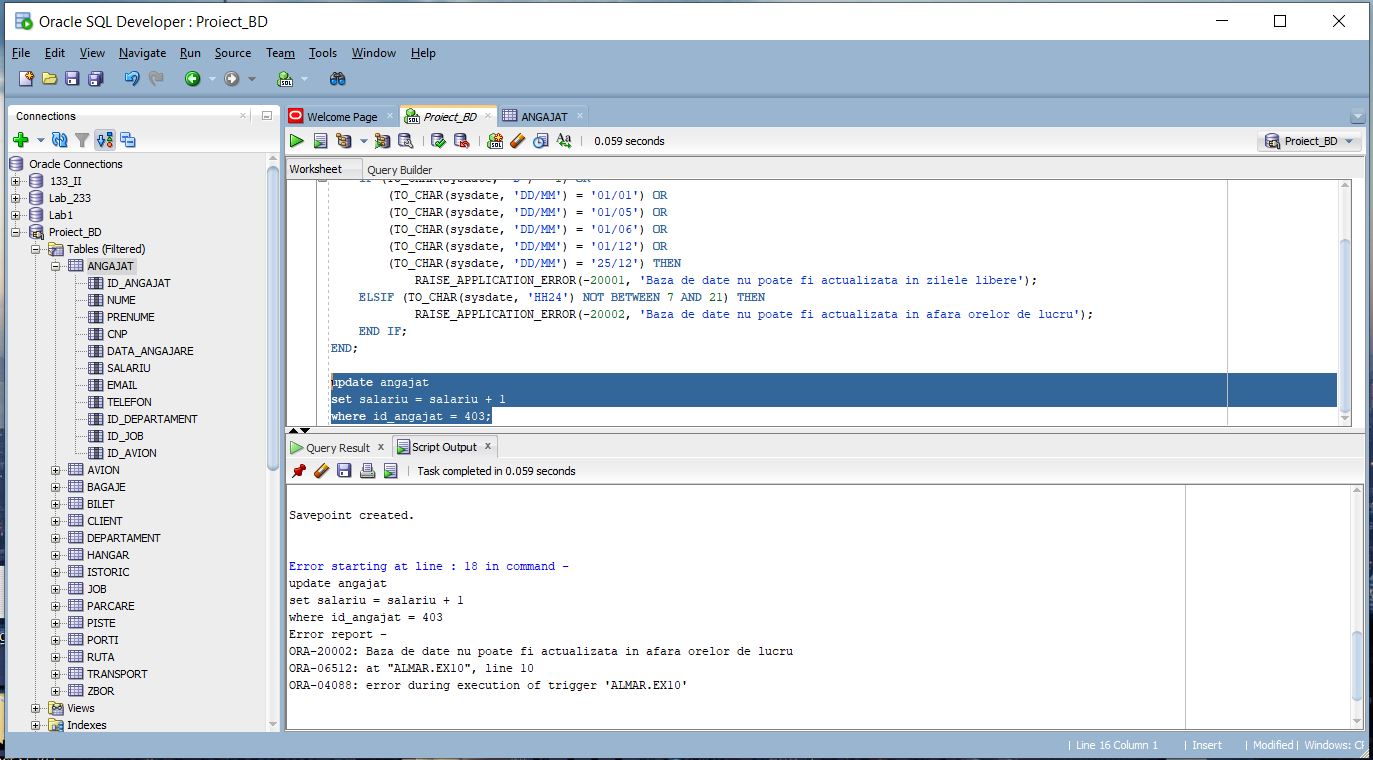
ELSIF (TO\_CHAR(sysdate, 'HH24') NOT BETWEEN 7 AND 21) THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Baza de date nu poate fi actualizata in afara orelor de lucru');

END IF;

END;

/



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

Definiti un trigger de tip LMD care sa nu permita modificarea pretului unui zbor la clasa a 2-a, astfel incat acesta sa devina mai mare decat cel la clasa 1.

CREATE OR REPLACE TRIGGER ex11

BEFORE UPDATE OF pret\_clasa2, pret\_clasa1 ON zbor

FOR EACH ROW

BEGIN

IF (:NEW.pret\_clasa2 > :OLD.pret\_clasa1) THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Pretul la clasa a 2-a e prea mare');

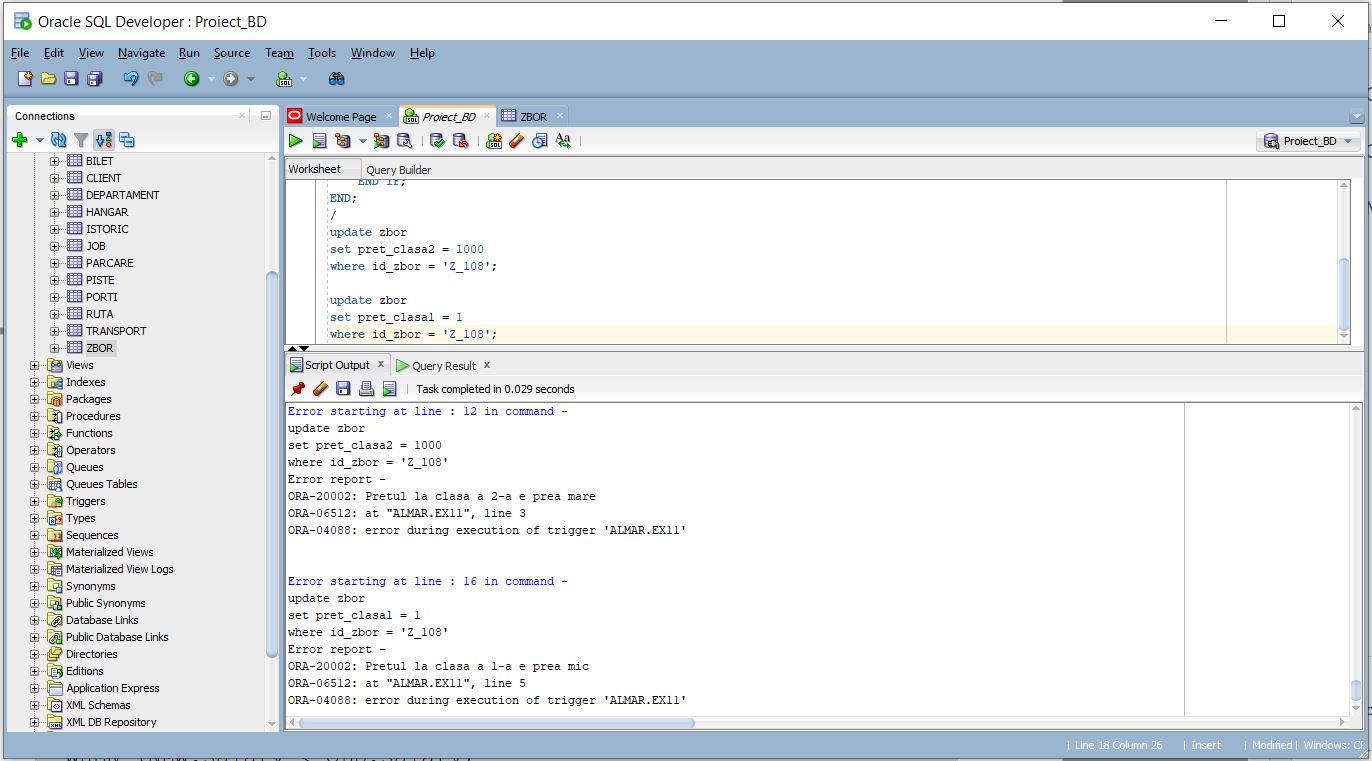
ELSIF (:NEW.pret\_clasa1 < :OLD.pret\_clasa2) THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Pretul la clasa a 1-a e prea mic');

END IF;

END;

/



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

Afisati un mesaj cu numele utilizatorului, comanda executata, obiectul modificat si data curenta, de fiecare data cand se executa o comanda de tip LDD.

CREATE OR REPLACE TRIGGER ex12

BEFORE CREATE OR DROP OR ALTER ON SCHEMA

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Comanda LDD noua: ');

DBMS\_OUTPUT.PUT\_LINE('Utilizatorul: ' || SYS.login\_user);

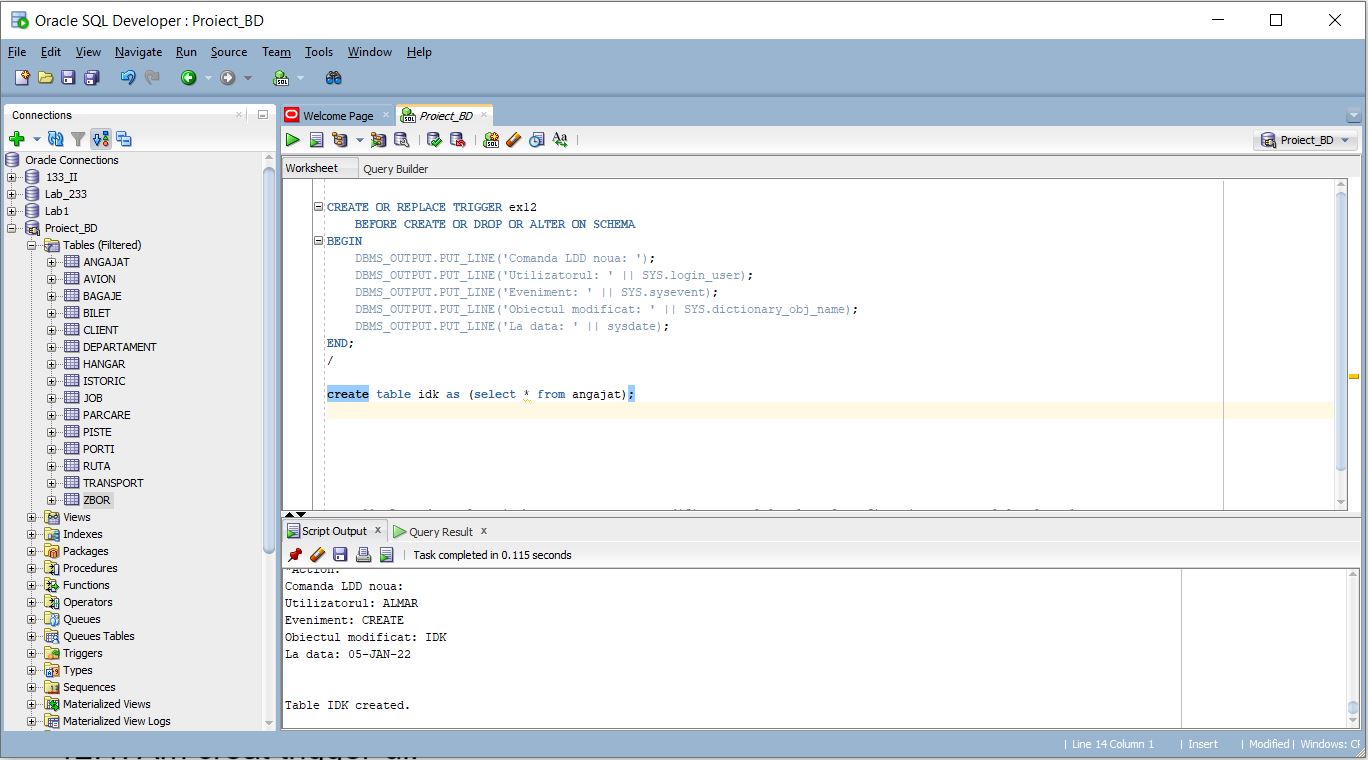
DBMS\_OUTPUT.PUT\_LINE('Eveniment: ' || SYS.sysevent);

DBMS\_OUTPUT.PUT\_LINE('Obiectul modificat: ' || SYS.dictionary\_obj\_name);

DBMS\_OUTPUT.PUT\_LINE('La data: ' || sysdate);

END;

/



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

CREATE OR REPLACE PACKAGE ex13 AS

PROCEDURE ex6;

PROCEDURE ex7;

FUNCTION ex8(dest ruta.destinatie%TYPE) RETURN avion.companie\_aeriana%TYPE;

PROCEDURE ex9(cl client.nume%TYPE);

END ex13;

/

CREATE OR REPLACE PACKAGE BODY ex13 AS

PROCEDURE ex6

IS

TYPE tablou\_indexat IS TABLE OF angajat.salariu%TYPE INDEX BY PLS\_INTEGER;

t1 tablou\_indexat;

TYPE tablou\_imbricat IS TABLE OF angajat.salariu%TYPE;

t2 tablou\_imbricat := tablou\_imbricat();

BEGIN

SELECT salariu

BULK COLLECT INTO t1

FROM angajat

WHERE id\_avion IS NULL;

SELECT salariu

BULK COLLECT INTO t2

FROM angajat

WHERE id\_avion IS NOT NULL;

DBMS\_OUTPUT.PUT\_LINE('Salariul nou al angajatilor care nu efectueaza zboruri:');

FOR i IN t1.first .. t1.last LOOP

DBMS\_OUTPUT.PUT\_LINE(0.05\*t1(i) + t1(i));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Salariul nou al angajatilor care efectueaza zboruri:');

FOR i IN t2.first .. t2.last LOOP

DBMS\_OUTPUT.PUT\_LINE(0.1\*t2(i) + t2(i));

END LOOP;

END ex6;

PROCEDURE ex7

IS

dest\_prec ruta.destinatie%TYPE := '0';

CURSOR c IS

SELECT ruta.destinatie NUME\_RUTA, zbor.id\_zbor ZBOR\_ID

FROM ruta, zbor

WHERE ruta.id\_ruta = zbor.id\_ruta

ORDER BY NUME\_RUTA;

BEGIN

FOR i IN c LOOP

IF dest\_prec = '0' OR dest\_prec != i.NUME\_RUTA THEN

DBMS\_OUTPUT.PUT\_LINE('Zboruri catre ' || i.NUME\_RUTA || ':');

DBMS\_OUTPUT.PUT\_LINE(i.ZBOR\_ID);

dest\_prec := i.NUME\_RUTA;

ELSE

DBMS\_OUTPUT.PUT\_LINE(i.ZBOR\_ID);

END IF;

END LOOP;

END ex7;

FUNCTION ex8

(dest ruta.destinatie%TYPE)

RETURN avion.companie\_aeriana%TYPE

IS

comp avion.companie\_aeriana%TYPE;

BEGIN

SELECT avion.companie\_aeriana

INTO comp

FROM avion, zbor, ruta

WHERE avion.id\_avion = zbor.id\_avion AND

zbor.id\_ruta = ruta.id\_ruta AND

ruta.destinatie = dest;

RETURN comp;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu sunt zboruri catre ruta selectata');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe zboruri catre ruta selectata');

WHEN OTHERS THEN

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

END ex8;

PROCEDURE ex9

(cl client.nume%TYPE)

IS

id\_z bilet.id\_zbor%TYPE;

data\_plec zbor.data\_plecare%TYPE;

dest ruta.destinatie%TYPE;

comp avion.companie\_aeriana%TYPE;

BEGIN

SELECT bilet.id\_zbor, zbor.data\_plecare, ruta.destinatie, avion.companie\_aeriana

INTO id\_z, data\_plec, dest, comp

FROM client, bilet, zbor, ruta, avion

WHERE client.id\_client = bilet.id\_client and

bilet.id\_zbor = zbor.id\_zbor and

zbor.id\_ruta = ruta.id\_ruta and

zbor.id\_avion = avion.id\_avion and

client.nume = cl;

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

DBMS\_OUTPUT.PUT\_LINE('ID zbor: ' || id\_z);

DBMS\_OUTPUT.PUT\_LINE('Data de plecare: ' || data\_plec);

DBMS\_OUTPUT.PUT\_LINE('Destinatia: ' || dest);

DBMS\_OUTPUT.PUT\_LINE('Compania aeriana: ' || comp);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista client cu acest nume');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul are mai mult de un bilet');

WHEN OTHERS THEN

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

END ex9;

END ex13;

/

BEGIN

ex13.ex6;

END;

/

BEGIN

ex13.ex7;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ex13.ex8('Dubai'));

END;

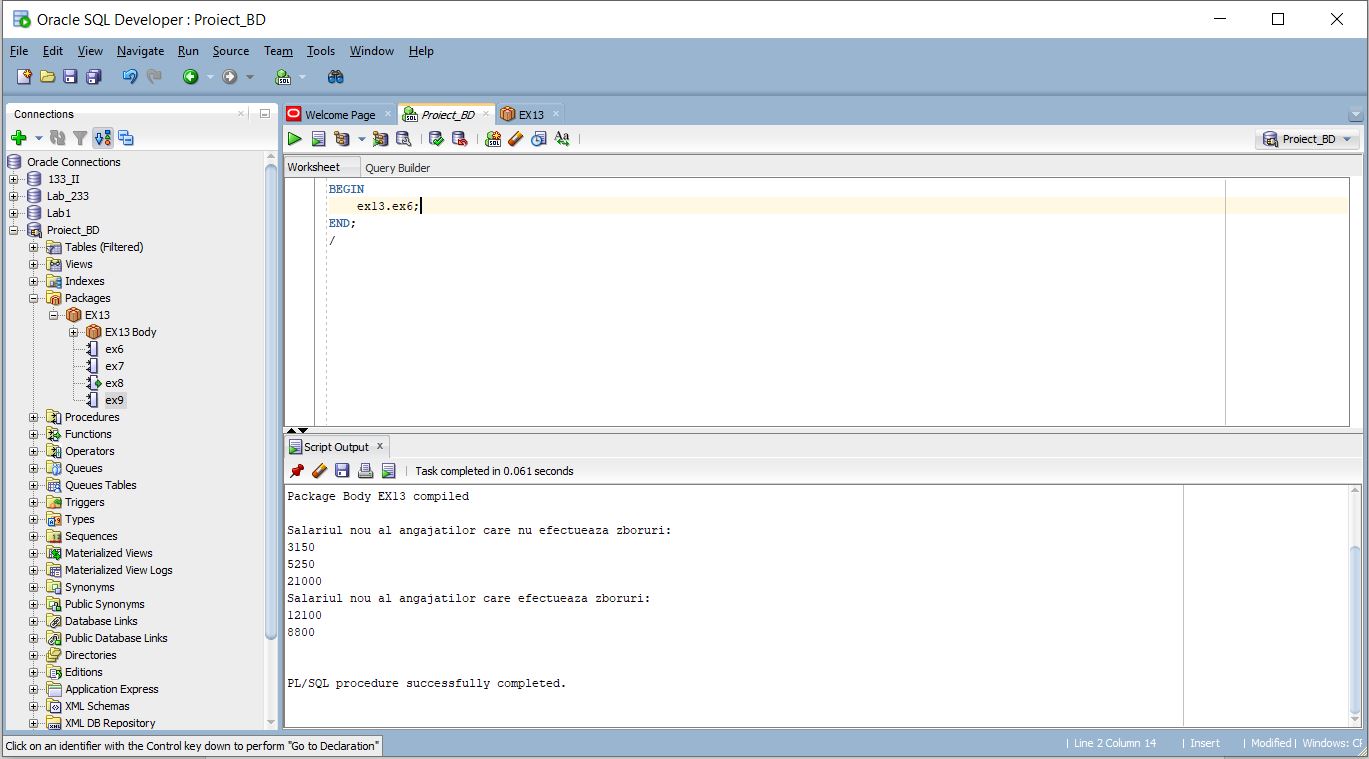
/

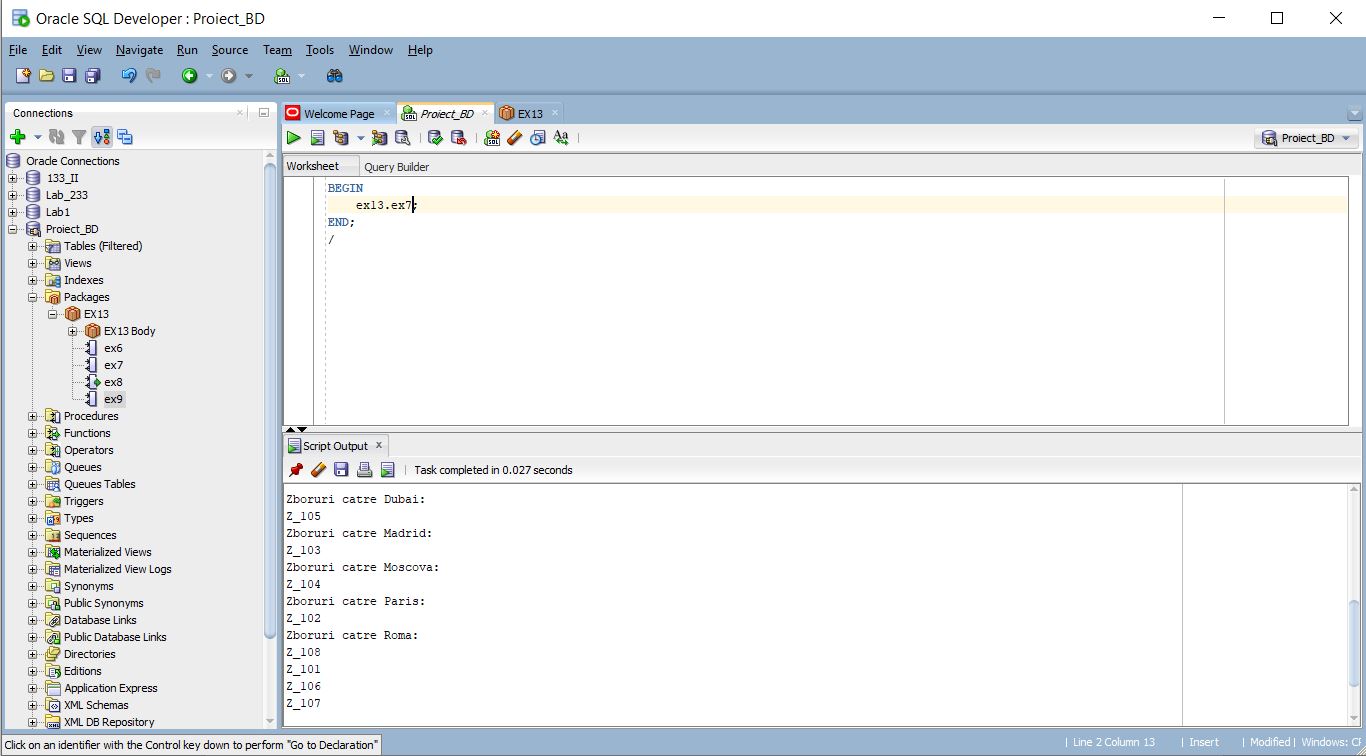
BEGIN

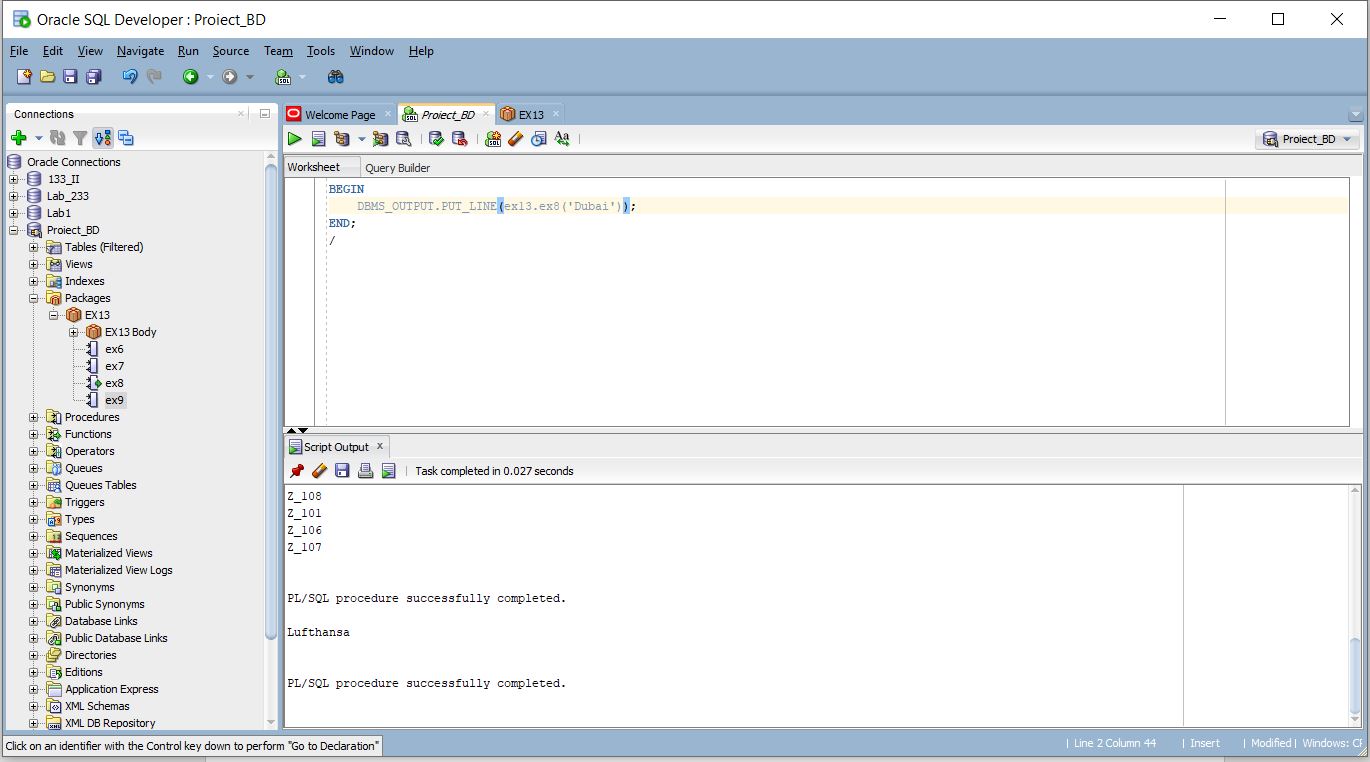
ex13.ex9('Ion');

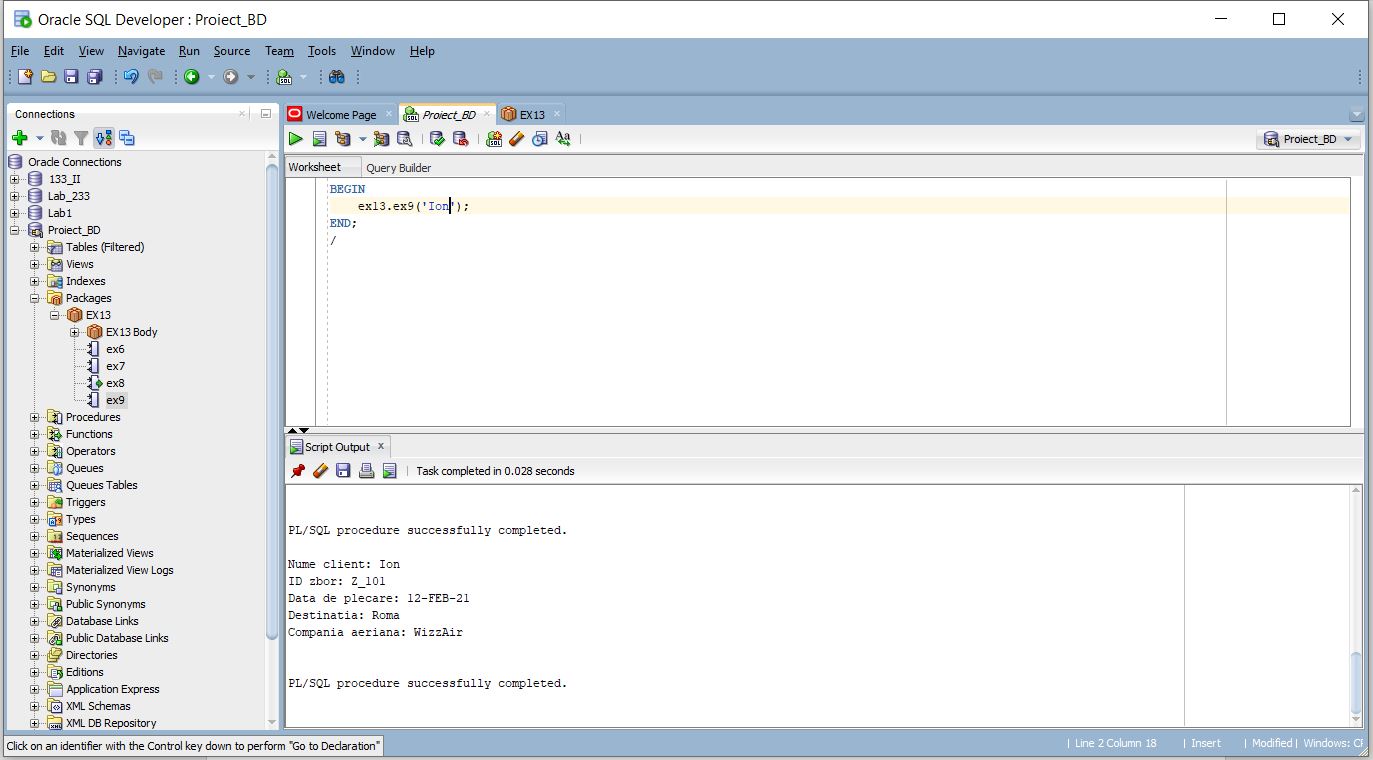
END;

/









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).**

Am creat un pachet care sa contina un record, un nested table, 2 functii si 2 proceduri. Prima functie returneaza un record in care se afla id-ul unui zbor dat ca si parametru si venitul total strans din vanzarea biletelor catre zborul selectat. A doua functie returneaza un nested table cu id-urile zborurilor catre o ruta data ca parametru. Prima procedura afiseaza totalul de venit pe ruta data ca parametru, folosindu-se de cele 2 functii, iar a doua procedura afiseaza pe ecran totalul pentru toate rutele existente.

CREATE OR REPLACE PACKAGE ex14 AS

TYPE total\_zbor IS RECORD

(id zbor.id\_zbor%TYPE,

total1 zbor.pret\_clasa1%TYPE,

total2 zbor.pret\_clasa2%TYPE);

TYPE zboruri IS TABLE OF zbor.id\_zbor%TYPE;

FUNCTION f\_total\_zbor(id\_z zbor.id\_zbor%TYPE) RETURN total\_zbor;

FUNCTION f\_zboruri(dest ruta.destinatie%TYPE) RETURN zboruri;

PROCEDURE p\_total\_ruta(citire\_ruta ruta.destinatie%TYPE);

PROCEDURE p\_total\_incasari;

END ex14;

/

CREATE OR REPLACE PACKAGE BODY ex14 AS

FUNCTION f\_total\_zbor(id\_z zbor.id\_zbor%TYPE) RETURN total\_zbor IS

tz total\_zbor;

BEGIN

SELECT zbor.id\_zbor, sum(zbor.pret\_clasa1), sum(zbor.pret\_clasa2)

INTO tz.id, tz.total1, tz.total2

FROM bilet, zbor

WHERE bilet.id\_zbor = zbor.id\_zbor and

zbor.id\_zbor = id\_z

GROUP BY zbor.id\_zbor;

RETURN tz;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

tz.id := id\_z;

tz.total1 := 0;

tz.total2 := 0;

RETURN tz;

END f\_total\_zbor;

FUNCTION f\_zboruri(dest ruta.destinatie%TYPE) RETURN zboruri IS

z zboruri := zboruri();

BEGIN

SELECT zbor.id\_zbor

BULK COLLECT INTO z

FROM zbor, ruta

WHERE ruta.id\_ruta = zbor.id\_ruta and

ruta.destinatie = dest;

RETURN z;

END f\_zboruri;

PROCEDURE p\_total\_ruta(citire\_ruta ruta.destinatie%TYPE) IS

z zboruri;

tz total\_zbor;

suma1 zbor.pret\_clasa1%TYPE := 0;

suma2 zbor.pret\_clasa2%TYPE := 0;

BEGIN

z := f\_zboruri(citire\_ruta);

FOR i IN z.first..z.last LOOP

tz := f\_total\_zbor(z(i));

suma1 := suma1 + tz.total1;

suma2 := suma2 + tz.total2;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Total profit pe ruta ' || citire\_ruta || ' la clasa 1: ' || suma1);

DBMS\_OUTPUT.PUT\_LINE('Total profit pe ruta ' || citire\_ruta || ' la clasa 2: ' || suma2);

END p\_total\_ruta;

PROCEDURE p\_total\_incasari IS

CURSOR c IS

SELECT destinatie d

FROM ruta;

BEGIN

FOR i IN c LOOP

p\_total\_ruta(i.d);

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

END LOOP;

END p\_total\_incasari;

END ex14;

/

Begin

ex14.p\_total\_incasari;

end;

