Baza de date a unui retele de aeroporturi

Realizat de: Iftimi Ruxandra

Cuprins

1. Descrierea modelului + Constrangeri (restrictii, reguli)
2. Diagrama Entitate-Relatie
3. Diagrama Conceptuala
4. Implementare Diagrama
5. Inserarea Datelor
6. Cerinta6 + Rezolvare
7. Cerinta7 + Rezolvare
8. Cerinta8 + Rezolvare
9. Cerinta9 + Rezolvare
10. Cerinta10 + Rezolvare
11. Cerinta11 + Rezolvare
12. Cerinta12 + Rezolvare
13. Cerinta13 + Rezolvare
14. Cerinta14 + Rezolvare

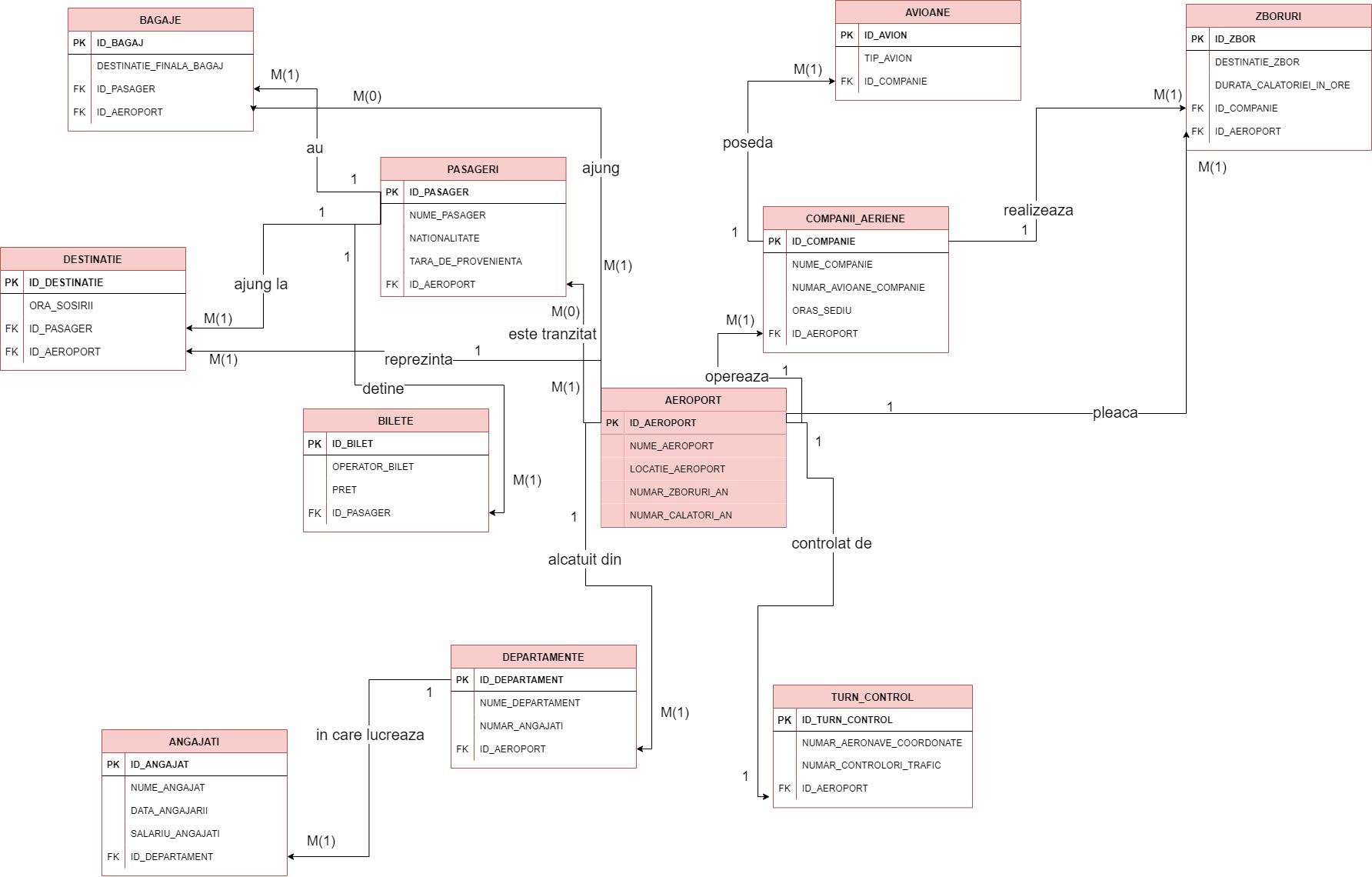
1. Prezentați pe scurt baza de date (utilitatea ei).

Pentru acest proiect, am ales realizarea bazei de date a unei retele de aeroporturi, formata din Aeroportul “Henri Coanda”, Aeroportul “Schphol”, Aeroportul “Charles des Gaulle”, Aeroportul ”Heathrow”, Aeroportul “John F. Kennedy” si Aeroportul “Beijing”. Aceasta are ca entitate principala “AEROPORT” si elementele sale principale componente: pasagerii ce tranziteaza aeroporturile, companiile aeriene ce opereaza pe acele aeroporturi, departamentele ce le alcatuiesc si turnurile lor de control. Pasagerii dispun de bilete individuale, cu bagajul inclus sau fara, si vor sa ajunga la destinatie. Acest lucru se poate realiza cu ajutorul companiilor aeriene care vand biletele de avion pentru zborurile lor, realizate cu avioane din flota acestora. Activitatea aeriana a unui aeroport se realizeaza cu ajutorul turnului sau de control ce se ocupa de desfasurarea in siguranta a zborurilor si coordonarea avioanelor de pe fiecare aeroport in parte. Totodata, la baza activitatii de pe fiecare aeroport, se afla departamentele sale si angajatii lor, menite sa organizeze activitatea interna a acestuia, astfel incat sa nu existe deficiente. Utilitatea principala a acestei baze de date consta in faptul ca, in orice moment, mai ales in cele de urgenta, se pot afla toate informatiile necesare pentru a se detecta problema.

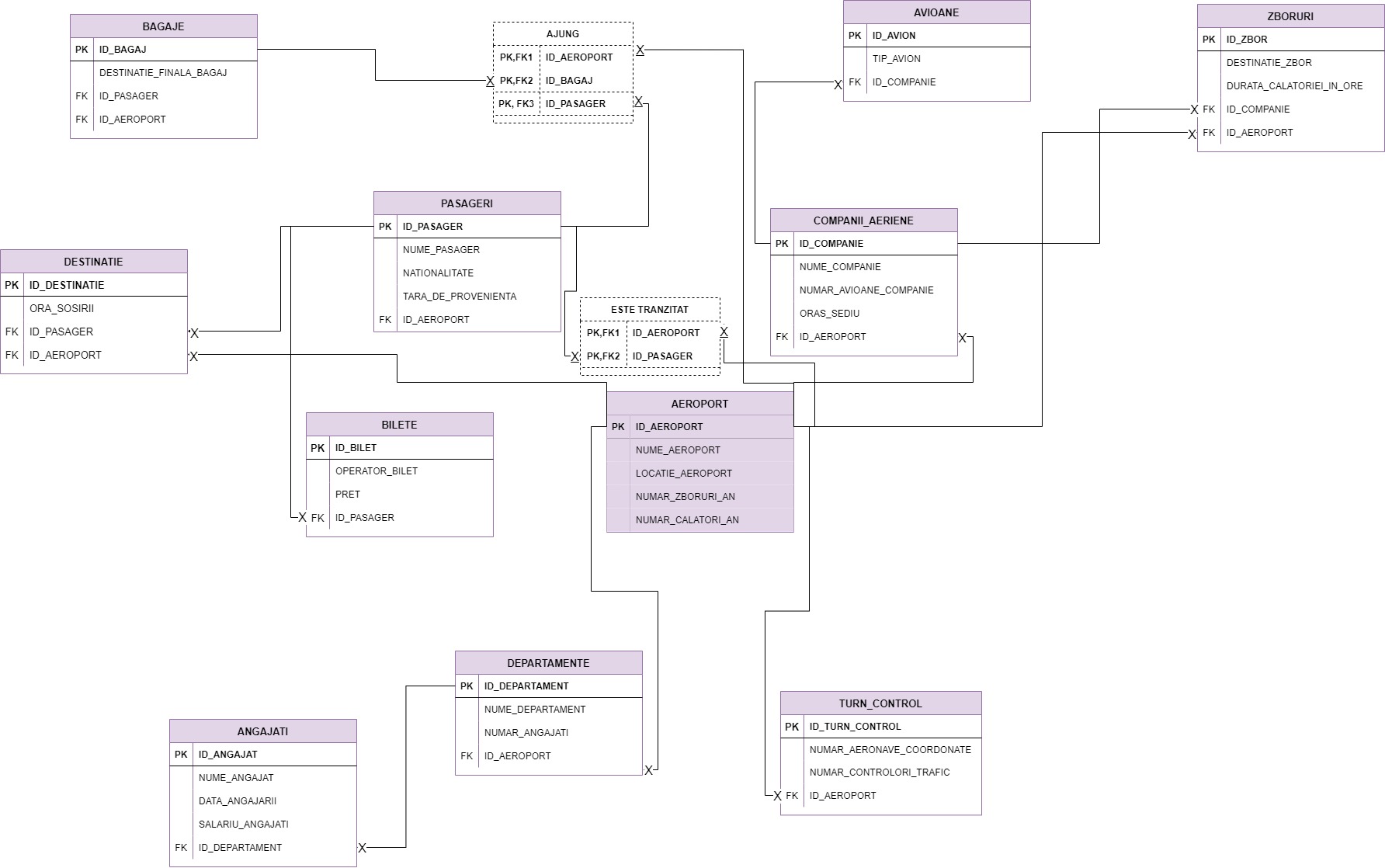
Restrictii si reguli:

* Departamentele difera in functie de fiecare aeroport
* Biletul poate fi vandut si prin intermediul unei firme/agentii de turism, nu neaparat de o companie aeriana
* Un pasager se identifica unic pentru fiecare zbor la care participa (pentru a nu se suprapune cu un eventual id al altui pasager al altui zbor)
* Fiecare pasager trebuie sa aiba un singur bagaj de cala (bagajele de mana nu intra in aceasta categorie, intrucat ele vor ajunge in cabina)
* Toate zborurile catre o anumita destinatie vor fi unice, intrucat ele vor depinde de identificarea lor din cadrul companiei aeriene si de id-ul acesteia
* Zborurile sunt unice pentru fiecare aeroport, intrucat depind de id-ul acestuia si id-ul companiei
* Companiile aeriene care opereaza pe un anumit aeroport sunt unice pentru acesta, intrucat isi desfasoara activitatea in functie de conditiile aeroportului(adica TAROM Otopeni are id diferit de TAROM Londra)
* Turnul de control este unic pentru fiecare aeroport
* Un aeroport este alcatuit din cel putin un department care are cel putin un angajat
* Pe un aeroport exista posibilitatea sa nu existe niciun pasager, dar sa existe zboruri de marfuri ce garanteaza activitatea aeroportului
* Pe un aeroport exista posibilitatea sa nu ajunga niciun bagaj daca nu ajunge niciun pasager
* Biletul de avion este unic pentru fiecare pasager si nu este transmisibil
* O companie aeriana trebuie sa aiba cel putin un zbor ca sa opereze
* Pasagerii pot sa aiba mai multe bilete (pentru escale)
* Biletul de avion poate fi si castigat, deci persoana care intra in posesia lui nu va plati

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 AEROPORT(

ID\_AEROPORT int not null primary key,

NUME\_AEROPORT varchar2(50),

LOCATIE\_AEROPORT varchar2(50),

NUMAR\_ZBORURI\_AN int,

NUMAR\_CALATORI\_AN long

);

--drop table AEROPORT;

create table DEPARTAMENTE(

ID\_DEPARTAMENT int not null primary key,

NUME\_DEPARTAMENT varchar2(30),

NUMAR\_ANGAJATI int,

ID\_AEROPORT not null,

foreign key (ID\_AEROPORT) references AEROPORT(ID\_AEROPORT)

);

--drop table DEPARTAMENTE;

create table ANGAJATI(

ID\_ANGAJAT int not null primary key,

NUME\_ANGAJAT varchar2(20),

DATA\_ANGAJARII varchar2(30),

SALARIU\_ANGAJATI int,

ID\_DEPARTAMENT not null,

foreign key(ID\_DEPARTAMENT) references DEPARTAMENTE(ID\_DEPARTAMENT)

);

--drop table ANGAJATI;

create table TURN\_CONTROL(

ID\_TURN\_CONTROL int not null primary key,

NUMAR\_AERONAVE\_COORDONATE long,

NUMAR\_CONTROLORI\_TRAFIC int,

ID\_AEROPORT not null,

foreign key(ID\_AEROPORT) references AEROPORT(ID\_AEROPORT)

);

--drop table TURN\_CONTROL;

create table COMPANII\_AERIENE(

ID\_COMPANIE int not null primary key,

NUME\_COMPANIE varchar2(50),

NUMAR\_AVION\_COMPANIE long,

ORAS\_SEDIU varchar2(30),

ID\_AEROPORT not null,

foreign key(ID\_AEROPORT) references AEROPORT(ID\_AEROPORT)

);

--drop table COMPANII\_AERIENE;

create table AVIOANE(

ID\_AVION int not null primary key,

TIP\_AVION varchar2(20),

ID\_COMPANIE not null,

foreign key(ID\_COMPANIE) references COMPANII\_AERIENE(ID\_COMPANIE)

);

--drop table AVIOANE;

create table ZBORURI(

ID\_ZBOR int not null primary key,

DESTINATIE\_ZBOR varchar2(30),

DURATA\_CALATORIEI\_IN\_ORE int,

ID\_AEROPORT not null,

ID\_COMPANIE not null,

foreign key(ID\_COMPANIE) references COMPANII\_AERIENE(ID\_COMPANIE),

foreign key(ID\_AEROPORT) references AEROPORT(ID\_AEROPORT)

);

--drop table ZBORURI;

create table PASAGERI (

ID\_PASAGER int not null primary key,

NUME\_PASAGER varchar2(20),

NATIONALITATE varchar2(30),

TARA\_DE\_PROVENIENTA varchar2(30),

ID\_AEROPORT not null,

foreign key(ID\_AEROPORT) references AEROPORT(ID\_AEROPORT)

);

--drop table PASAGERI;

create table BILETE (

ID\_BILET int not null primary key,

OPERATOR\_BILET varchar2(50),

PRET int,

ID\_PASAGER not null,

foreign key(ID\_PASAGER) references PASAGERI(ID\_PASAGER)

);

--drop table BILETE;

create table BAGAJE (

ID\_BAGAJ int not null primary key,

DESTINATIE\_FINALA\_BAGAJ varchar2(30),

ID\_AEROPORT not null,

ID\_PASAGER not null,

foreign key(ID\_AEROPORT) references AEROPORT(ID\_AEROPORT),

foreign key(ID\_PASAGER) references PASAGERI(ID\_PASAGER)

);

--drop table BAGAJE;

create table DESTINATIE (

ID\_DESTINATIE int not null primary key,

ORA\_SOSIRII int,

ID\_AEROPORT not null,

ID\_PASAGER not null,

foreign key(ID\_AEROPORT) references AEROPORT(ID\_AEROPORT),

foreign key(ID\_PASAGER) references PASAGERI(ID\_PASAGER)

);

--drop table DESTINATIE;

create table ESTE\_TRANZITAT (

ID\_PASAGER int not null,

ID\_AEROPORT int not null,

constraint pk\_este\_tranzitat primary key(ID\_PASAGER, ID\_AEROPORT)

);

-- drop table ESTE\_TRANZITAT;

create table AJUNG (

ID\_PASAGER int not null,

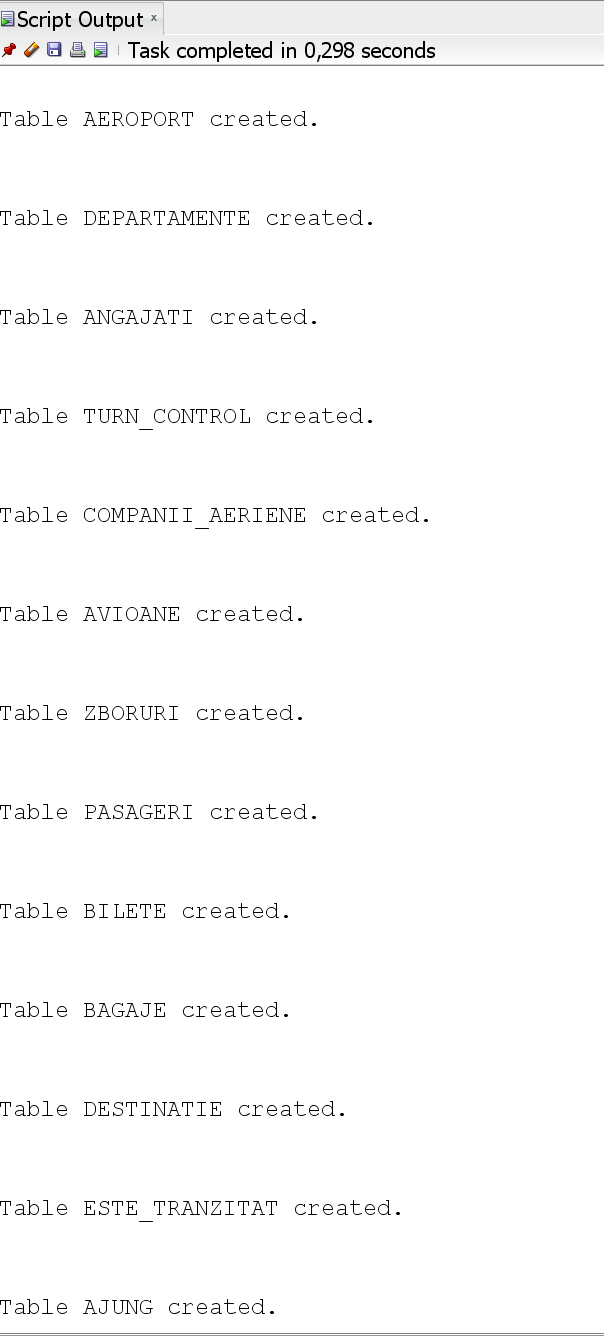
ID\_AEROPORT int not null,

ID\_BAGAJ int not null,

constraint pk\_ajung primary key(ID\_PASAGER, ID\_AEROPORT, ID\_BAGAJ)

);

--drop table AJUNG;



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

insert into AEROPORT values (1000, 'Henri Coanda', 'Bucuresti', 500, 20000);

insert into AEROPORT values (1100, 'Schipol', 'Amsterdam', 1000, 100500);

insert into AEROPORT values (1200, 'Heathrow', 'Londra', 1500, 300000);

insert into AEROPORT values (1300, 'Charles de Gaulle', 'Paris', 1100, 100000);

insert into AEROPORT values (1400, 'John F. Kennedy', 'New York', 2000, 900000);

insert into AEROPORT values (1500, 'Beijing', 'Beijing', 3000, 1000000);



insert into DEPARTAMENTE values (1001, 'HR' ,100, 1000);

insert into DEPARTAMENTE values (1002, 'PR' ,100, 1000);

insert into DEPARTAMENTE values (1003, 'Logistica', 50, 1100);

insert into DEPARTAMENTE values (1004, 'Comercial', 40, 1100);

insert into DEPARTAMENTE values (1005, 'Marketing', 25, 1200);

insert into DEPARTAMENTE values (1006, 'Control', 10, 1200);

insert into DEPARTAMENTE values (1007, 'Securitate', 5, 1300);

insert into DEPARTAMENTE values (1008, 'Operatiuni', 80, 1300);

insert into DEPARTAMENTE values (1009, 'Management', 45, 1400);

insert into DEPARTAMENTE values (1010, 'Achizitii', 30, 1400);

insert into DEPARTAMENTE values (1011, 'Finante', 100, 1500);

insert into DEPARTAMENTE values (1012, 'Dezvoltare', 30, 1500);

insert into DEPARTAMENTE values (1013, 'Director\_1000', 5, 1000);

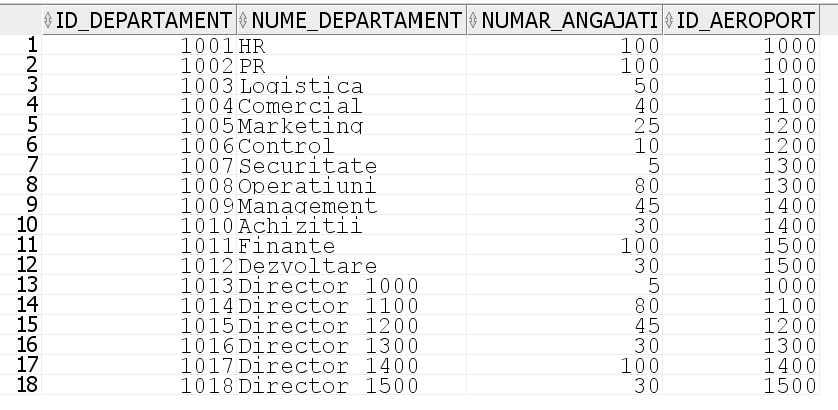
insert into DEPARTAMENTE values (1014, 'Director\_1100', 80, 1100);

insert into DEPARTAMENTE values (1015, 'Director\_1200', 45, 1200);

insert into DEPARTAMENTE values (1016, 'Director\_1300', 30, 1300);

insert into DEPARTAMENTE values (1017, 'Director\_1400', 100, 1400);

insert into DEPARTAMENTE values (1018, 'Director\_1500', 30, 1500);



insert into ANGAJATI values (111, 'Popescu', '21-03-2010', 2000, 1001);

insert into ANGAJATI values (112, 'Marinescu', '18-09-2009', 5000, 1002);

insert into ANGAJATI values (113, 'Popa', '20-05-2011', 4500, 1003);

insert into ANGAJATI values (114, 'Spataru', '03-03-2010', 6300, 1004);

insert into ANGAJATI values (115, 'Nastase', '04-05-2003', 5200, 1005);

insert into ANGAJATI values (116, 'Afrim', '01-08-2013', 3900, 1006);

insert into ANGAJATI values (117, 'Alexa', '15-12-2015', 4800, 1007);

insert into ANGAJATI values (118, 'Ionita', '21-10-2012', 6700, 1008);

insert into ANGAJATI values (119, 'Bercu', '17-02-2005', 5400, 1009);

insert into ANGAJATI values (120, 'Nita', '18-08-2013', 5500, 1010);

insert into ANGAJATI values (121, 'Plesa', '27-06-2014', 5500, 1011);

insert into ANGAJATI values (122, 'Duta', '05-07-2004', 7200, 1012);

insert into ANGAJATI values (123, 'Hobbs', '03-03-2010', 8900, 1013);

insert into ANGAJATI values (124, 'Sargent', '21-09-2012', 9200, 1014);

insert into ANGAJATI values (125, 'Galloway', '02-02-2006', 10000, 1015);

insert into ANGAJATI values (126, 'Connolly', '04-08-2013', 9500, 1016);

insert into ANGAJATI values (127, 'Holcomb', '05-06-2014', 9800, 1017);

insert into ANGAJATI values (128, 'Sheehan', '08-09-2004', 8800, 1018);

insert into ANGAJATI values (129, 'Thornton', '09-08-1995', 8330, 1001);

insert into ANGAJATI values (130, 'Villegas', '03-01-1996', 9400, 1002);

insert into ANGAJATI values (131, 'Stanton', '30-06-1997', 8770, 1003);

insert into ANGAJATI values (132, 'Taylor', '18-09-1997', 4100, 1004);

insert into ANGAJATI values (133, 'Bates', '14-11-1997', 8630, 1005);

insert into ANGAJATI values (134, 'Ingram', '01-06-2000', 7400, 1006);

insert into ANGAJATI values (135, 'Carey', '14-08-2000', 8900, 1007);

insert into ANGAJATI values (136, 'Patton', '18-04-2003', 2550, 1008);

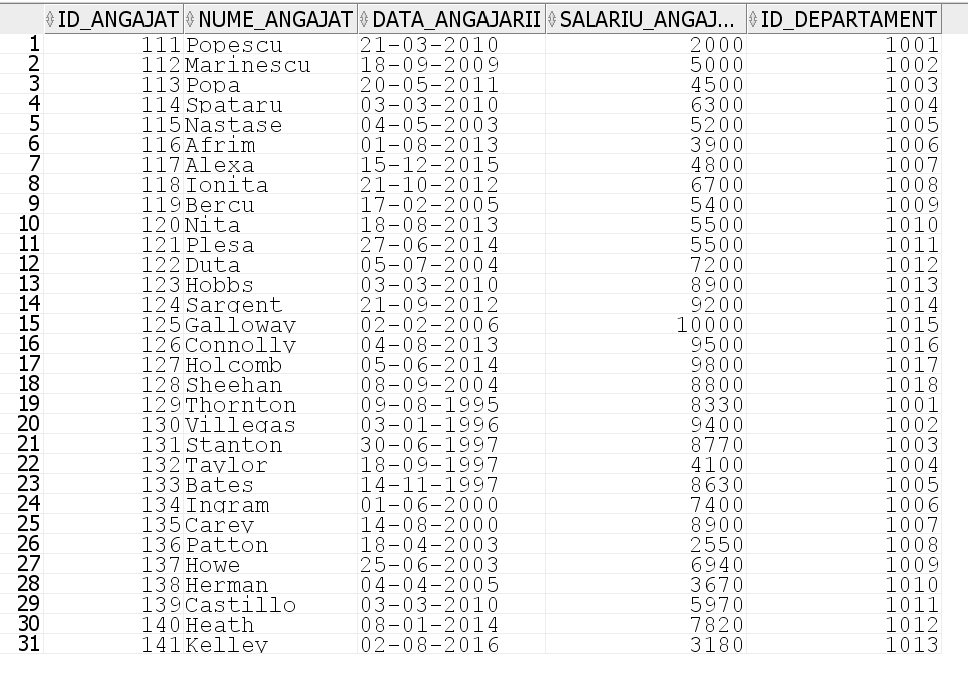
insert into ANGAJATI values (137, 'Howe', '25-06-2003', 6940, 1009);

insert into ANGAJATI values (138, 'Herman', '04-04-2005', 3670, 1010);

insert into ANGAJATI values (139, 'Castillo', '03-03-2010', 5970, 1011);

insert into ANGAJATI values (140, 'Heath', '08-01-2014', 7820, 1012);

insert into ANGAJATI values (141, 'Kelley', '02-08-2016', 3180, 1013);



insert into TURN\_CONTROL values (900, 600, 30, 1000);

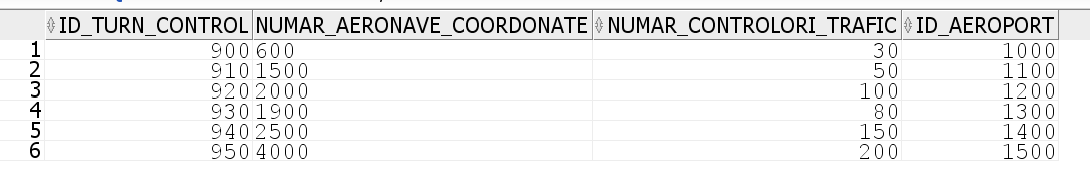
insert into TURN\_CONTROL values (910, 1500, 50, 1100);

insert into TURN\_CONTROL values (920, 2000, 100, 1200);

insert into TURN\_CONTROL values (930, 1900, 80, 1300);

insert into TURN\_CONTROL values (940, 2500, 150, 1400);

insert into TURN\_CONTROL values (950, 4000, 200, 1500);



insert into PASAGERI values (90, 'Popa', 'Romana', 'Germania', 1000);

insert into PASAGERI values (91, 'Phan', 'Chineza', 'Japonia', 1500);

insert into PASAGERI values (92, 'Edwards', 'Irlandeza', 'Argentina', 1200);

insert into PASAGERI values (93, 'Bass', 'Britanica', 'Egipt', 1100);

insert into PASAGERI values (94, 'Senior', 'Canadiana', 'Ungaria', 1300);

insert into PASAGERI values (95, 'Mcclain', 'Americana', 'Finlanda', 1400);

insert into PASAGERI values (100, 'King', 'Romana', 'Italia', 1000);

insert into PASAGERI values (101, 'Liu', 'Chineza', 'Australia', 1500);

insert into PASAGERI values (102, 'Lott', 'Britanica', 'SUA', 1200);

insert into PASAGERI values (103, 'Bishop', 'Canadiana', 'Africa de Sud', 1100);

insert into PASAGERI values (104, 'Medina', 'Italiana', 'Franta', 1300);

insert into PASAGERI values (105, 'Franks', 'Americana', 'Ucraina', 1400);



insert into BILETE values (500, 'TAROM', 100, 100);

insert into BILETE values (501, 'KLM', 300, 101);

insert into BILETE values (502, 'British Airways', 200, 102);

insert into BILETE values (503, 'AirChina', 500, 103);

insert into BILETE values (504, 'Delta', 700, 104);

insert into BILETE values (505, 'AirFrance', 400, 105);

insert into BILETE values (506, 'TAROM', 200, 90);

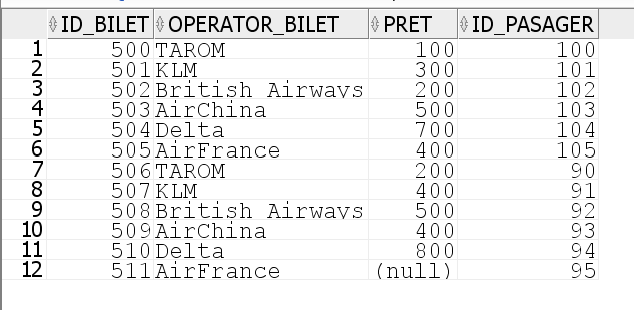
insert into BILETE values (507, 'KLM', 400, 91);

insert into BILETE values (508, 'British Airways', 500, 92);

insert into BILETE values (509, 'AirChina', 400, 93);

insert into BILETE values (510, 'Delta', 800, 94);

insert into BILETE values (511, 'AirFrance', null, 95);



insert into BAGAJE values (10, 'Bucuresti', 1000, 100);

insert into BAGAJE values (11, 'Beijing', 1500, 101);

insert into BAGAJE values (12, 'Londra', 1200, 102);

insert into BAGAJE values (13, 'Amsterdam', 1100, 103);

insert into BAGAJE values (14, 'Paris', 1300, 104);

insert into BAGAJE values (15, 'New York', 1400, 105);

insert into BAGAJE values (16, 'Bucuresti', 1000, 90);

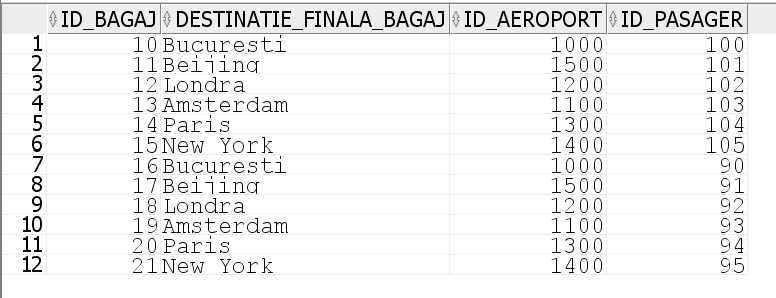
insert into BAGAJE values (17, 'Beijing', 1500, 91);

insert into BAGAJE values (18, 'Londra', 1200, 92);

insert into BAGAJE values (19, 'Amsterdam', 1100, 93);

insert into BAGAJE values (20, 'Paris', 1300, 94);

insert into BAGAJE values (21, 'New York', 1400, 95);



insert into DESTINATIE values (3000, 20, 1000, 100);

insert into DESTINATIE values (3100, 15, 1500, 101);

insert into DESTINATIE values (3200, 01, 1200, 102);

insert into DESTINATIE values (3300, 10, 1100, 103);

insert into DESTINATIE values (3400, 19, 1300, 104);

insert into DESTINATIE values (3500, 12, 1400, 105);

insert into DESTINATIE values (3001, 07, 1000, 90);

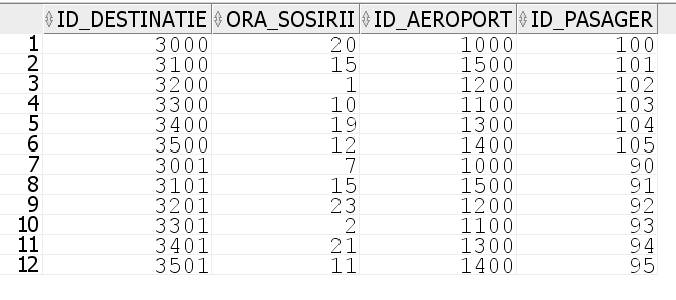
insert into DESTINATIE values (3101, 15, 1500, 91);

insert into DESTINATIE values (3201, 23, 1200, 92);

insert into DESTINATIE values (3301, 02, 1100, 93);

insert into DESTINATIE values (3401, 21, 1300, 94);

insert into DESTINATIE values (3501, 11, 1400, 95);



insert into COMPANII\_AERIENE values (800, 'TAROM', 50, 'Bucuresti', 1000);

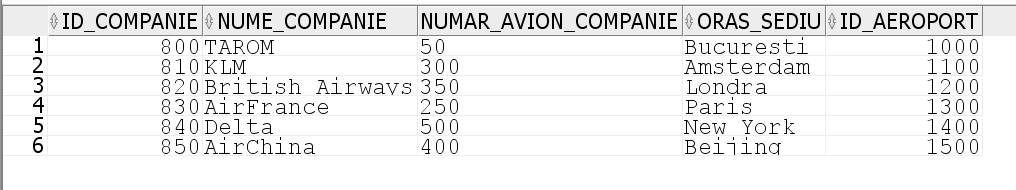
insert into COMPANII\_AERIENE values (810, 'KLM', 300, 'Amsterdam', 1100);

insert into COMPANII\_AERIENE values (820, 'British Airways', 350, 'Londra', 1200);

insert into COMPANII\_AERIENE values (830, 'AirFrance', 250, 'Paris', 1300);

insert into COMPANII\_AERIENE values (840, 'Delta', 500, 'New York', 1400);

insert into COMPANII\_AERIENE values (850, 'AirChina', 400, 'Beijing', 1500);



insert into AVIOANE values (50, 'Boeing 707', 800);

insert into AVIOANE values (51, 'Boeing 787', 800);

insert into AVIOANE values (52, 'Airbus A300', 810);

insert into AVIOANE values (53, 'Airbus A340', 810);

insert into AVIOANE values (54, 'Airbus A310', 820);

insert into AVIOANE values (55, 'Airbus A330', 820);

insert into AVIOANE values (56, 'Boeing 717', 830);

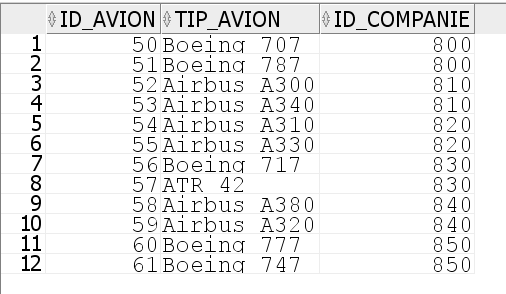
insert into AVIOANE values (57, 'ATR 42', 830);

insert into AVIOANE values (58, 'Airbus A380', 840);

insert into AVIOANE values (59, 'Airbus A320', 840);

insert into AVIOANE values (60, 'Boeing 777', 850);

insert into AVIOANE values (61, 'Boeing 747', 850);



insert into ZBORURI values (5000, 'Bucuresti', 2, 1000, 800);

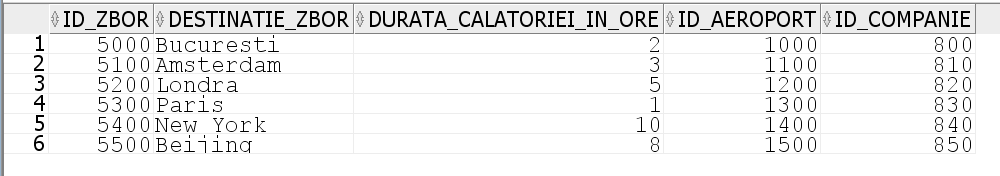
insert into ZBORURI values (5100, 'Amsterdam', 3, 1100, 810);

insert into ZBORURI values (5200, 'Londra', 5, 1200, 820);

insert into ZBORURI values (5300, 'Paris', 1, 1300, 830);

insert into ZBORURI values (5400, 'New York', 10, 1400, 840);

insert into ZBORURI values (5500, 'Beijing', 8, 1500, 850);



insert into AJUNG values (1000, 10, 100);

insert into AJUNG values (1100, 11, 101);

insert into AJUNG values (1200, 12, 102);

insert into AJUNG values (1300, 13, 103);

insert into AJUNG values (1400, 14, 104);

insert into AJUNG values (1500, 15, 105);

insert into AJUNG values (1000, 16, 90);

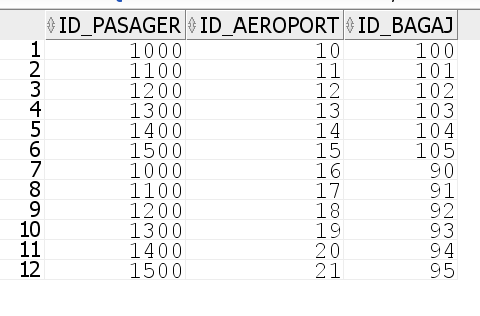
insert into AJUNG values (1100, 17, 91);

insert into AJUNG values (1200, 18, 92);

insert into AJUNG values (1300, 19, 93);

insert into AJUNG values (1400, 20, 94);

insert into AJUNG values (1500, 21, 95);



insert into ESTE\_TRANZITAT values (1000, 10);

insert into ESTE\_TRANZITAT values (1100, 11);

insert into ESTE\_TRANZITAT values (1200, 12);

insert into ESTE\_TRANZITAT values (1300, 13);

insert into ESTE\_TRANZITAT values (1400, 14);

insert into ESTE\_TRANZITAT values (1500, 15);

insert into ESTE\_TRANZITAT values (1000, 16);

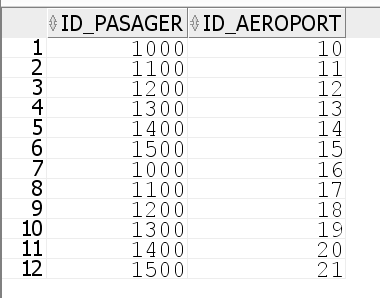
insert into ESTE\_TRANZITAT values (1100, 17);

insert into ESTE\_TRANZITAT values (1200, 18);

insert into ESTE\_TRANZITAT values (1300, 19);

insert into ESTE\_TRANZITAT values (1400, 20);

insert into ESTE\_TRANZITAT values (1500, 21);



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.

/\* Pentru fiecare departament de pe un aeroport cu numele dat, sa se afiseze toti angajatii

care nu sunt directori in acel departament si daca acestia au salariul mai mare decat media

salariilor din acel departament.\*/

--am folosit ca si colectii: tablouri indexate (table\_dep) si tablouri imbricate (ta)

create or replace procedure exercitiul6\_ir (a aeroport.nume\_aeroport%type) as

type tablou\_dep is table of departamente.nume\_departament%type index by pls\_integer;

t tablou\_dep;

v\_id aeroport.id\_aeroport%type;

v\_director departamente.id\_departament%type;

type tablou\_angajati is table of angajati.nume\_angajat%type;

ta tablou\_angajati := tablou\_angajati();

v\_salariu angajati.salariu\_angajati%type;

numar int;

v\_medie int;

begin

select id\_aeroport

into v\_id

from aeroport

where upper(nume\_aeroport)=upper(a);

select id\_departament

into v\_director

from departamente

where v\_id=id\_aeroport and nume\_departament like 'Director%';

select nume\_departament

bulk collect into t

from departamente

where v\_id=id\_aeroport and id\_departament != v\_director;

for i in t.first..t.last loop

select round(avg(salariu\_angajati))

into v\_medie

from angajati a, departamente dep

where a.id\_departament=dep.id\_departament and dep.nume\_departament=t(i);

DBMS\_OUTPUT.PUT('In departamentul ' || t(i) || ' cu salariul mediu ' || v\_medie || ' lucreaza angajatii: ');

DBMS\_OUTPUT.NEW\_LINE;

select a.nume\_angajat

bulk collect into ta

from angajati a, departamente dep

where a.id\_departament=dep.id\_departament and dep.nume\_departament=t(i);

numar:=1;

for j in ta.first..ta.last loop

DBMS\_OUTPUT.PUT(numar || '. ' || ta(j) || ', avand salariul ');

numar := numar +1;

select salariu\_angajati

into v\_salariu

from angajati

where ta(j)=nume\_angajat;

DBMS\_OUTPUT.PUT(v\_salariu || ' care este mai ');

if v\_salariu >= v\_medie then

DBMS\_OUTPUT.PUT('mare decat media salariilor din departament.');

else

DBMS\_OUTPUT.PUT('mic decat media salariilor din departament.');

end if;

DBMS\_OUTPUT.NEW\_LINE;

end loop;

DBMS\_OUTPUT.NEW\_LINE;

end loop;

end exercitiul6\_ir;

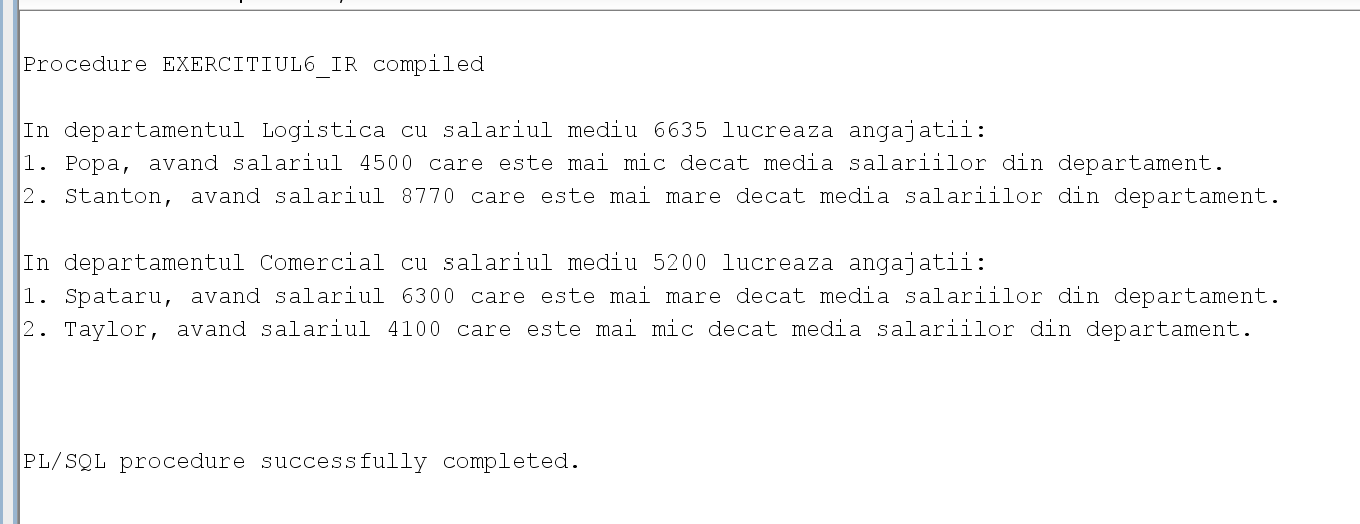
/

begin

exercitiul6\_ir('Schipol');

end;

/



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.

/\*Afisati fiecare pasager american numele, operatorul biletului sau (biletul are operator/ biletul a fost castigat),

pretul acestuia, cat si ora sosirii si orasul si aeroportul unde ajung bagajele sale.\*/

--am folosit ca si cursoare: ciclu cursor (c1) si cursor explicit parametrizat (c2)

create or replace procedure exercitiul7\_ir(nat pasageri.nationalitate%type) as

v\_nume varchar2(10);

type tablou\_pasageri is table of pasageri.nume\_pasager%type;

t tablou\_pasageri;

v\_id pasageri.id\_pasager%type;

v\_operator bilete.operator\_bilet%type;

v\_pret bilete.pret%type;

v\_ora destinatie.ora\_sosirii%type;

v\_destinatie bagaje.destinatie\_finala\_bagaj%type;

v\_aeroport aeroport.nume\_aeroport%type;

cursor c1 is

select nume\_pasager

from pasageri

where upper(nationalitate)=upper(nat);

cursor c2(pid pasageri.id\_pasager%type) is

select bil.operator\_bilet, bil.pret, d.ora\_sosirii, b.destinatie\_finala\_bagaj, a.nume\_aeroport

from bilete bil, destinatie d, bagaje b, aeroport a, pasageri p

where pid=bil.id\_pasager and pid=d.id\_pasager and pid=b.id\_pasager and

d.id\_aeroport=a.id\_aeroport and b.id\_aeroport=a.id\_aeroport and

p.id\_aeroport=a.id\_aeroport;

begin

DBMS\_OUTPUT.PUT\_LINE('Cetatenii americani care tranziteaza reteaua de aeroporturi sunt: ');

DBMS\_OUTPUT.NEW\_LINE;

for i in c1 loop

DBMS\_OUTPUT.PUT(i.nume\_pasager || ' ');

select id\_pasager

into v\_id

from pasageri

where upper(nume\_pasager)=upper(i.nume\_pasager);

open c2(v\_id);

fetch c2 into v\_operator, v\_pret, v\_ora, v\_destinatie, v\_aeroport;

if v\_pret is null then

DBMS\_OUTPUT.PUT\_LINE('care a castigat biletul de avion vandut de ' || v\_operator || ' si va ateriza la ora ' || v\_ora || ' pe aeroportul ' || v\_aeroport || ' din orasul ' || v\_destinatie);

else

DBMS\_OUTPUT.PUT\_LINE('care a cumparat biletul de avion vandut de ' || v\_operator || ' cu pretul de ' || v\_pret || ' si va ateriza la ora ' || v\_ora || ' pe aeroportul ' || v\_aeroport || ' din orasul ' || v\_destinatie);

end if;

close c2;

--DBMS\_OUTPUT.NEW\_LINE;

end loop;

end exercitiul7\_ir;

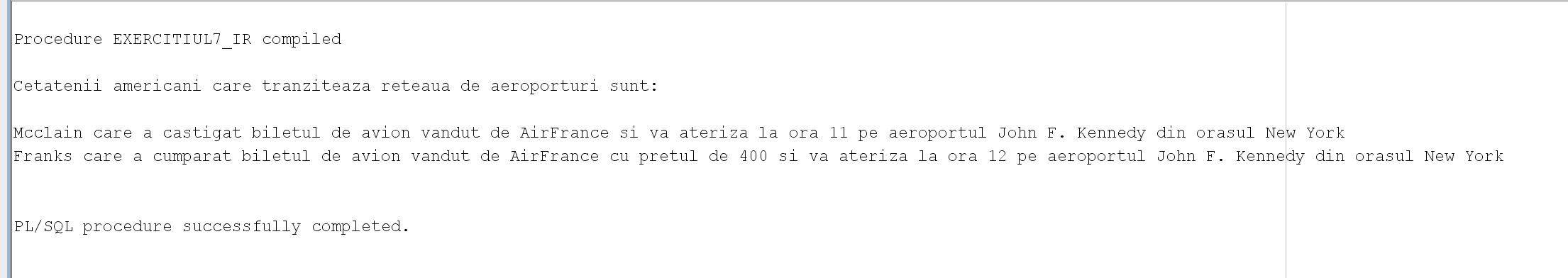
/

begin

exercitiul7\_ir('americana');

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.

/\*Pentru id-ul unui turn de control dat, aflati care este este numele companiei care

opereaza pe aeroporul cu turnul de control respectiv si detine avioane care contin in

numele lor "bus" si destinatia zborurilor operate de companie este "AMSTERDAM". Se

vor trata erorile: daca codul este invalid si daca nu exista companii care sa

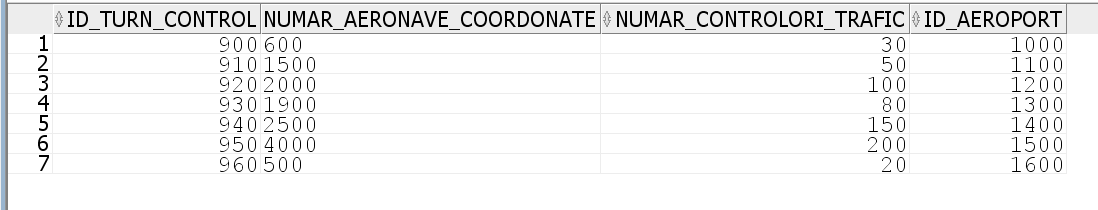
indeplineasca cerintele\*/

--mai adaug niste date

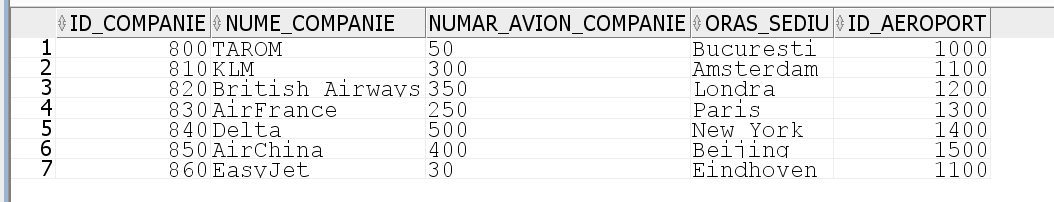
insert into AEROPORT values (1600, 'El Prat', 'Barcelona', 900, 90000);



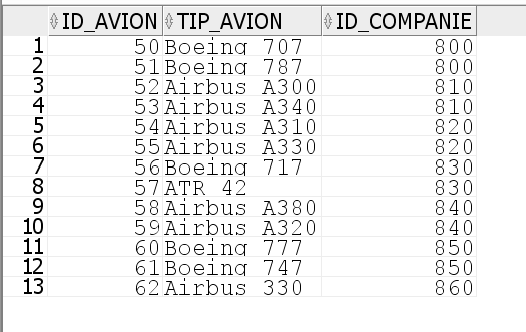
insert into TURN\_CONTROL values (960, 500, 20, 1600);



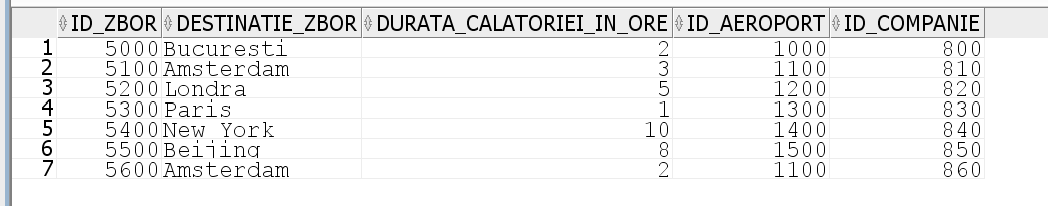
insert into COMPANII\_AERIENE values (860, 'EasyJet', 30, 'Eindhoven', 1100);



insert into AVIOANE values (62, 'Airbus 330', 860);



insert into ZBORURI values (5600, 'Amsterdam', 2, 1100, 860);



create or replace function exercitiul8\_ir(tid turn\_control.id\_turn\_control%type) return varchar2 is

e1 exception;

e2 exception;

--v\_nume companii\_aeriene.nume\_companie%type;

v\_id aeroport.id\_aeroport%type;

type tablou is table of companii\_aeriene.nume\_companie%type;

t tablou;

begin

if tid<900 or tid>999 then

raise e1;

end if;

select id\_aeroport

into v\_id

from turn\_control

where id\_turn\_control=tid;

--DBMS\_OUTPUT.PUT\_LINE(v\_id);

select c.nume\_companie

bulk collect into t

from companii\_aeriene c

where v\_id=c.id\_aeroport and c.id\_companie in ( select id\_companie

from avioane

where tip\_avion like '%bus%') and c.id\_companie in (select id\_companie

from zboruri

where upper(destinatie\_zbor)='AMSTERDAM'

);

if t.count()<1 then

raise e2;

else

return('Id valid');

end if;

exception

when e1 then return('Id invalid');

when e2 then return('Nu se indeplinesc conditiile.');

when others then return('Cod de eroare: ' || sqlcode);

end exercitiul8\_ir;

/

begin

DBMS\_OUTPUT.PUT\_LINE(exercitiul8\_ir(800));

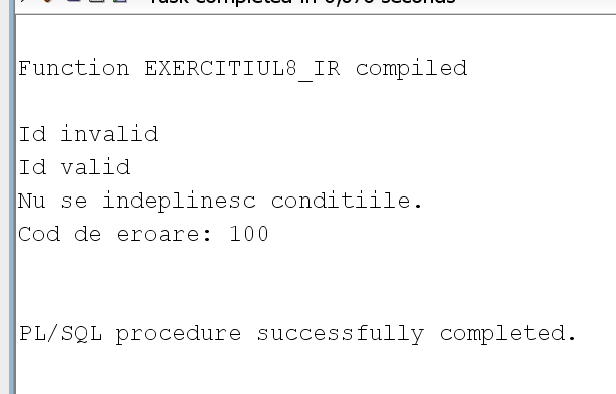
DBMS\_OUTPUT.PUT\_LINE(exercitiul8\_ir(910));

DBMS\_OUTPUT.PUT\_LINE(exercitiul8\_ir(960));

DBMS\_OUTPUT.PUT\_LINE(exercitiul8\_ir(970));

end;

/



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 ora de sosire a pasagerului cu un nume dat a carui bilet este

vandut de operatorul care opereaza pe aeroportul cu locatia "BUCURESTI". Sa se

trateze cazurile in care exista mai multi pasageri cu acelasi nume sau nu exista deloc.\*/

--mai adaug niste date

insert into PASAGERI values (106, 'Edwards', 'Franceza', 'Brazilia', 1200);

insert into PASAGERI values (107, 'Edwards', 'Austriaca', 'Spania', 1300);



create or replace procedure exercitiul9\_ir(pnume pasageri.nume\_pasager%type) as

e1 exception;

e2 exception;

type tablou\_nume is table of pasageri.nume\_pasager%type;

t tablou\_nume;

v\_ora varchar2(10);

begin

select nume\_pasager

bulk collect into t

from pasageri

where nume\_pasager=pnume;

if sql%notfound then

raise e2;

end if;

if t.count()>=2 then

raise e1;

end if;

for i in t.first..t.last loop

select d.ora\_sosirii

into v\_ora

from destinatie d, pasageri p

where p.nume\_pasager=t(i) and p.id\_pasager=d.id\_pasager and p.id\_pasager in ( select id\_pasager

from bilete

where operator\_bilet in ( select nume\_companie

from companii\_aeriene

where id\_aeroport in ( select id\_aeroport

from aeroport

where upper(locatie\_aeroport)='BUCURESTI'

)));

DBMS\_OUTPUT.PUT\_LINE('Ora sosirii pasagerului ' || t(i) || ' este ' || v\_ora);

end loop;

exception

when e1 then DBMS\_OUTPUT.PUT\_LINE('Exista mai multi calatori cu acest nume');

when e2 then DBMS\_OUTPUT.PUT\_LINE('Nu exista calatori cu acest nume');

when others then DBMS\_OUTPUT.PUT\_LINE('Cod de eroare: ' || sqlcode);

end exercitiul9\_ir;

/

begin

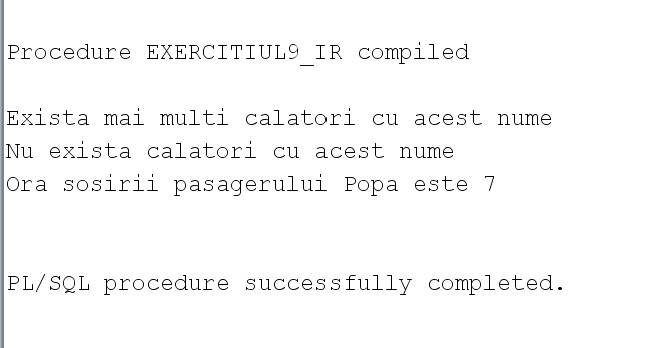
exercitiul9\_ir('Edwards');

exercitiul9\_ir('Iftimi');

exercitiul9\_ir('Popa');

end;

/



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

/\*Nu se poate angaja personal in luna ianuarie\*/

create or replace trigger exercitiul10\_ir

before insert or update or delete on angajati

begin

if(to\_char(sysdate, 'DD-MM')>'01-01' and to\_char(sysdate, 'DD-MM')<'31-01')

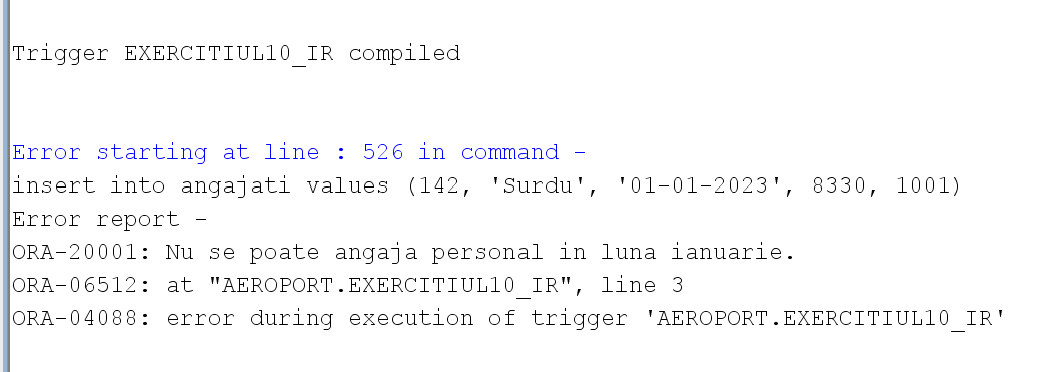
then raise\_application\_error(-20001,'Nu se poate angaja personal in luna ianuarie.');

end if;

end;

/

insert into angajati values (142, 'Surdu', '01-01-2023', 8330, 1001);



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

/\*Numarul persoanelor dintr-un departament nu poate fi marit.\*/

create or replace trigger exercitiul11\_ir

before update of numar\_angajati on departamente

for each row

declare

begin

if (:new.numar\_angajati > :old.numar\_angajati)

then raise\_application\_error(-20001,'Departamentele nu suporta atat de multi angajati.');

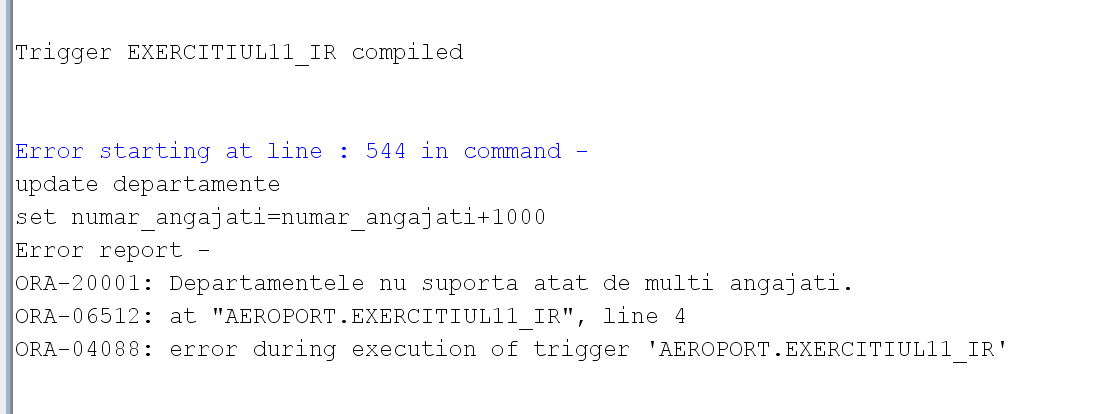
end if;

end;

/

update departamente

set numar\_angajati=numar\_angajati+1000;



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

create table ir\_ir(

utilizator varchar2(30),

nume\_bd varchar2(50),

eveniment varchar2(20),

nume\_obiect varchar2(30),

data date

);

create or replace trigger exercitiul12\_ir

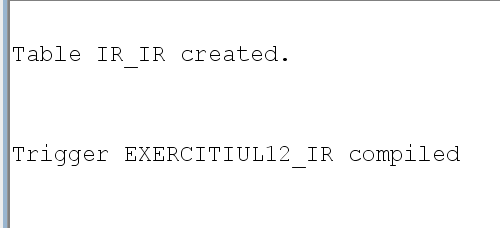
after create or drop or alter on schema

begin

insert into ir\_ir values(sys.login\_user, sys.database\_name, sys.sysevent, sys.dictionary\_obj\_name, sysdate);

end;

/



create table magazine (

id\_magazin int not null primary key,

nume\_magazin varchar2(50)

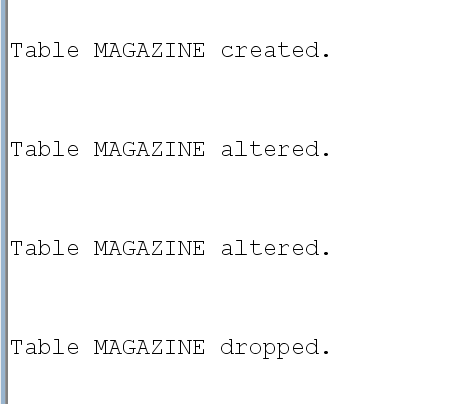
);

select \* from magazine;

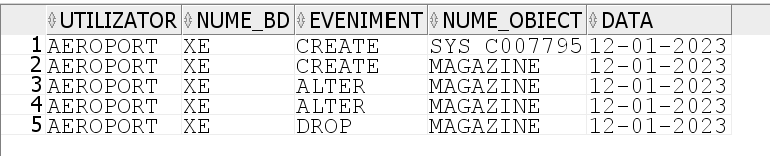
alter table magazine add (locatie varchar2(50));

alter table magazine drop column locatie;

drop table magazine;



select \* from ir\_ir;



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

create or replace package exercitiul13\_ir as

procedure exercitiul6\_ir (a aeroport.nume\_aeroport%type); --6

procedure exercitiul7\_ir(nat pasageri.nationalitate%type); --7

function exercitiul8\_ir(tid turn\_control.id\_turn\_control%type) return varchar2; --8

procedure exercitiul9\_ir(pnume pasageri.nume\_pasager%type); --9

end exercitiul13\_ir;

/

create or replace package body exercitiul13\_ir as

procedure exercitiul6\_ir (a aeroport.nume\_aeroport%type) as

type tablou\_dep is table of departamente.nume\_departament%type index by pls\_integer;

t tablou\_dep;

v\_id aeroport.id\_aeroport%type;

v\_director departamente.id\_departament%type;

type tablou\_angajati is table of angajati.nume\_angajat%type;

ta tablou\_angajati := tablou\_angajati();

v\_salariu angajati.salariu\_angajati%type;

numar int;

v\_medie int;

begin

select id\_aeroport

into v\_id

from aeroport

where upper(nume\_aeroport)=upper(a);

select id\_departament

into v\_director

from departamente

where v\_id=id\_aeroport and nume\_departament like 'Director%';

select nume\_departament

bulk collect into t

from departamente

where v\_id=id\_aeroport and id\_departament != v\_director;

for i in t.first..t.last loop

select round(avg(salariu\_angajati))

into v\_medie

from angajati a, departamente dep

where a.id\_departament=dep.id\_departament and dep.nume\_departament=t(i);

DBMS\_OUTPUT.PUT('In departamentul ' || t(i) || ' cu salariul mediu ' || v\_medie || ' lucreaza angajatii: ');

DBMS\_OUTPUT.NEW\_LINE;

select a.nume\_angajat

bulk collect into ta

from angajati a, departamente dep

where a.id\_departament=dep.id\_departament and dep.nume\_departament=t(i);

numar:=1;

for j in ta.first..ta.last loop

DBMS\_OUTPUT.PUT(numar || '. ' || ta(j) || ', avand salariul ');

numar := numar +1;

select salariu\_angajati

into v\_salariu

from angajati

where ta(j)=nume\_angajat;

DBMS\_OUTPUT.PUT(v\_salariu || ' care este mai ');

if v\_salariu >= v\_medie then

DBMS\_OUTPUT.PUT('mare decat media salariilor din departament.');

else

DBMS\_OUTPUT.PUT('mic decat media salariilor din departament.');

end if;

DBMS\_OUTPUT.NEW\_LINE;

end loop;

DBMS\_OUTPUT.NEW\_LINE;

end loop;

end exercitiul6\_ir;

procedure exercitiul7\_ir(nat pasageri.nationalitate%type) as

v\_nume varchar2(10);

type tablou\_pasageri is table of pasageri.nume\_pasager%type;

t tablou\_pasageri;

v\_id pasageri.id\_pasager%type;

v\_operator bilete.operator\_bilet%type;

v\_pret bilete.pret%type;

v\_ora destinatie.ora\_sosirii%type;

v\_destinatie bagaje.destinatie\_finala\_bagaj%type;

v\_aeroport aeroport.nume\_aeroport%type;

cursor c1 is

select nume\_pasager

from pasageri

where upper(nationalitate)=upper(nat);

cursor c2(pid pasageri.id\_pasager%type) is

select bil.operator\_bilet, bil.pret, d.ora\_sosirii, b.destinatie\_finala\_bagaj, a.nume\_aeroport

from bilete bil, destinatie d, bagaje b, aeroport a, pasageri p

where pid=bil.id\_pasager and pid=d.id\_pasager and pid=b.id\_pasager and

d.id\_aeroport=a.id\_aeroport and b.id\_aeroport=a.id\_aeroport and

p.id\_aeroport=a.id\_aeroport;

begin

DBMS\_OUTPUT.PUT\_LINE('Cetatenii americani care tranziteaza reteaua de aeroporturi sunt: ');

DBMS\_OUTPUT.NEW\_LINE;

for i in c1 loop

DBMS\_OUTPUT.PUT(i.nume\_pasager || ' ');

select id\_pasager

into v\_id

from pasageri

where upper(nume\_pasager)=upper(i.nume\_pasager);

open c2(v\_id);

fetch c2 into v\_operator, v\_pret, v\_ora, v\_destinatie, v\_aeroport;

if v\_pret is null then

DBMS\_OUTPUT.PUT\_LINE('care a castigat biletul de avion vandut de ' || v\_operator || ' si va ateriza la ora ' || v\_ora || ' pe aeroportul ' || v\_aeroport || ' din orasul ' || v\_destinatie);

else

DBMS\_OUTPUT.PUT\_LINE('care a cumparat biletul de avion vandut de ' || v\_operator || ' cu pretul de ' || v\_pret || ' si va ateriza la ora ' || v\_ora || ' pe aeroportul ' || v\_aeroport || ' din orasul ' || v\_destinatie);

end if;

close c2;

--DBMS\_OUTPUT.NEW\_LINE;

end loop;

end exercitiul7\_ir;

function exercitiul8\_ir(tid turn\_control.id\_turn\_control%type) return varchar2 is

e1 exception;

e2 exception;

--v\_nume companii\_aeriene.nume\_companie%type;

v\_id aeroport.id\_aeroport%type;

type tablou is table of companii\_aeriene.nume\_companie%type;

t tablou;

begin

if tid<900 or tid>999 then

raise e1;

end if;

select id\_aeroport

into v\_id

from turn\_control

where id\_turn\_control=tid;

--DBMS\_OUTPUT.PUT\_LINE(v\_id);

select c.nume\_companie

bulk collect into t

from companii\_aeriene c

where v\_id=c.id\_aeroport and c.id\_companie in ( select id\_companie

from avioane

where tip\_avion like '%bus%') and c.id\_companie in (select id\_companie

from zboruri

where upper(destinatie\_zbor)='AMSTERDAM'

);

if t.count()<1 then

raise e2;

else

return('Id valid');

end if;

exception

when e1 then return('Id invalid');

when e2 then return('Nu se indeplinesc conditiile.');

when others then return('Cod de eroare: ' || sqlcode);

end exercitiul8\_ir;

procedure exercitiul9\_ir(pnume pasageri.nume\_pasager%type) as

e1 exception;

e2 exception;

type tablou\_nume is table of pasageri.nume\_pasager%type;

t tablou\_nume;

v\_ora varchar2(10);

begin

select nume\_pasager

bulk collect into t

from pasageri

where nume\_pasager=pnume;

if sql%notfound then

raise e2;

end if;

if t.count()>=2 then

raise e1;

end if;

for i in t.first..t.last loop

select d.ora\_sosirii

into v\_ora

from destinatie d, pasageri p

where p.nume\_pasager=t(i) and p.id\_pasager=d.id\_pasager and p.id\_pasager in ( select id\_pasager

from bilete

where operator\_bilet in ( select nume\_companie

from companii\_aeriene

where id\_aeroport in ( select id\_aeroport

from aeroport

where upper(locatie\_aeroport)='BUCURESTI'

)));

DBMS\_OUTPUT.PUT\_LINE('Ora sosirii pasagerului ' || t(i) || ' este ' || v\_ora);

end loop;

exception

when e1 then DBMS\_OUTPUT.PUT\_LINE('Exista mai multi calatori cu acest nume');

when e2 then DBMS\_OUTPUT.PUT\_LINE('Nu exista calatori cu acest nume');

when others then DBMS\_OUTPUT.PUT\_LINE('Cod de eroare: ' || sqlcode);

end exercitiul9\_ir;

end exercitiul13\_ir;

/

begin

exercitiul13\_ir.exercitiul6\_ir('Schipol');

dbms\_output.new\_line();

exercitiul13\_ir.exercitiul7\_ir('americana');

dbms\_output.new\_line();

DBMS\_OUTPUT.PUT\_LINE(exercitiul13\_ir.exercitiul8\_ir(800));

DBMS\_OUTPUT.PUT\_LINE(exercitiul13\_ir.exercitiul8\_ir(910));

DBMS\_OUTPUT.PUT\_LINE(exercitiul13\_ir.exercitiul8\_ir(960));

DBMS\_OUTPUT.PUT\_LINE(exercitiul13\_ir.exercitiul8\_ir(970));

dbms\_output.new\_line();

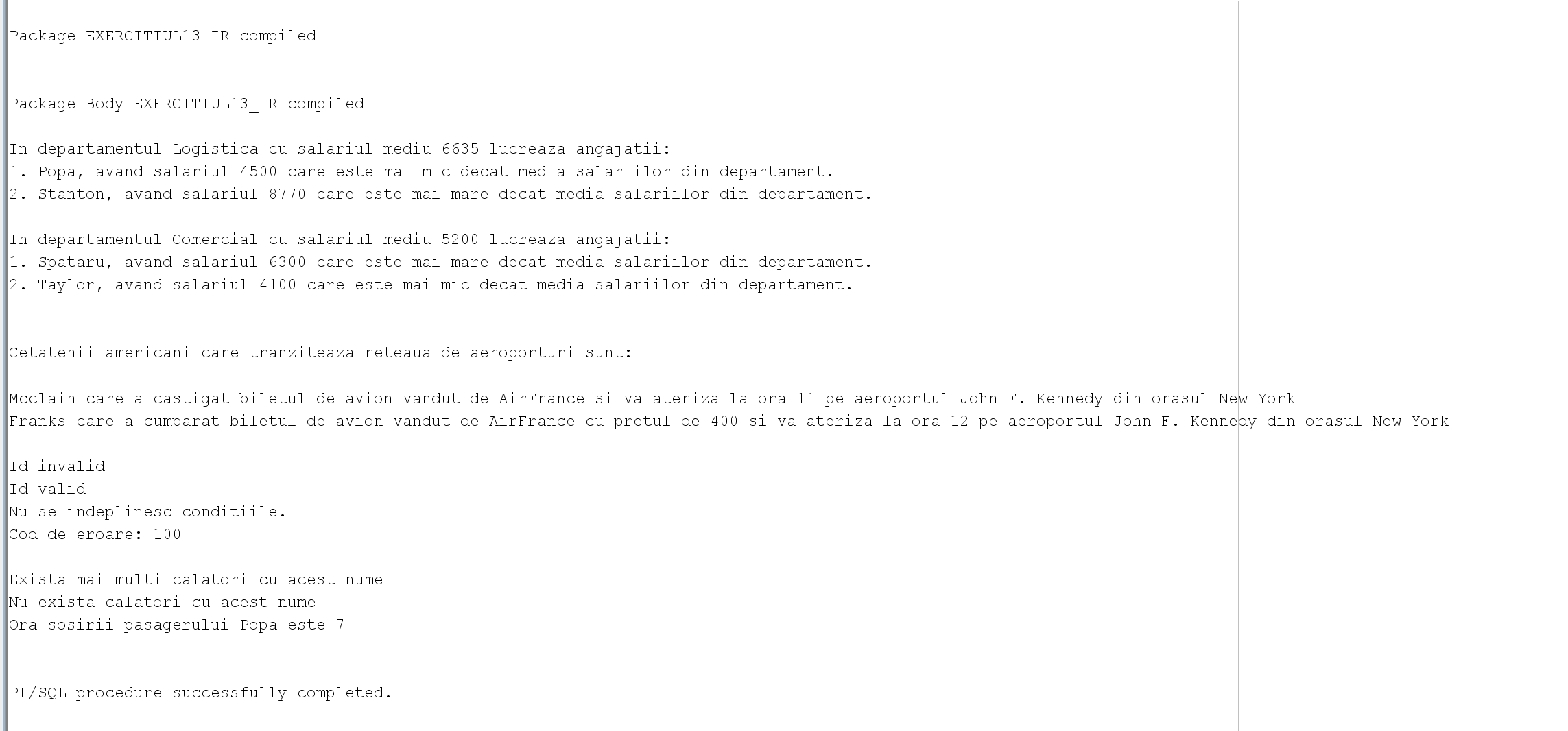
exercitiul13\_ir.exercitiul9\_ir('Edwards');

exercitiul13\_ir.exercitiul9\_ir('Iftimi');

exercitiul13\_ir.exercitiul9\_ir('Popa');

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).

create or replace package exercitiul14\_ir as

type tabel\_dep is table of departamente.nume\_departament%type index by pls\_integer;

t tabel\_dep;

type tablou\_angajati is table of angajati.nume\_angajat%type;

ta tablou\_angajati := tablou\_angajati();

function p\_can(a aeroport.id\_aeroport%type) return varchar2;

function p\_bid(i pasageri.id\_pasager%type) return number;

procedure p\_aid (a in aeroport.id\_aeroport%type);

procedure p\_pid (a in aeroport.id\_aeroport%type, n in pasageri.nume\_pasager%type);

end exercitiul14\_ir;

/

create or replace package body exercitiul14\_ir as

function p\_can(a aeroport.id\_aeroport%type) return varchar2 as

rez varchar2(100);

begin

select nume\_companie

into rez

from companii\_aeriene

where id\_aeroport=a;

return rez;

end p\_can;

function p\_bid(i pasageri.id\_pasager%type) return number as

rez number;

begin

select id\_bilet

into rez

from bilete

where id\_pasager=i;

return rez;

end p\_bid;

procedure p\_aid (a in aeroport.id\_aeroport%type) as

rez varchar2(100);

numar int;

v\_medie int;

v\_salariu angajati.salariu\_angajati%type;

begin

select nume\_aeroport

into rez

from aeroport

where a=id\_aeroport;

DBMS\_OUTPUT.PUT\_LINE('Aeroportul cu id-ul ' || a || ' si numele: ' || rez || ' are ca departamente: ');

select nume\_departament

bulk collect into t

from departamente

where a=id\_aeroport;

for i in t.first..t.last loop

select round(avg(salariu\_angajati))

into v\_medie

from angajati a, departamente dep

where a.id\_departament=dep.id\_departament and dep.nume\_departament=t(i);

DBMS\_OUTPUT.PUT(t(i) || ' cu salariul mediu ' || v\_medie || ' lucreaza angajatii: ');

DBMS\_OUTPUT.NEW\_LINE;

select a.nume\_angajat

bulk collect into ta

from angajati a, departamente dep

where a.id\_departament=dep.id\_departament and dep.nume\_departament=t(i);

numar:=1;

for j in ta.first..ta.last loop

DBMS\_OUTPUT.PUT(numar || '. ' || ta(j) || ', avand salariul ');

numar := numar +1;

select salariu\_angajati

into v\_salariu

from angajati

where ta(j)=nume\_angajat;

DBMS\_OUTPUT.PUT(v\_salariu || ' care este mai ');

if v\_salariu >= v\_medie then

DBMS\_OUTPUT.PUT('mare decat media salariilor din departament.');

else

DBMS\_OUTPUT.PUT('mic decat media salariilor din departament.');

end if;

DBMS\_OUTPUT.NEW\_LINE;

end loop;

DBMS\_OUTPUT.NEW\_LINE;

end loop;

end p\_aid;

procedure p\_pid (a in aeroport.id\_aeroport%type, n in pasageri.nume\_pasager%type) as

rez number;

begin

select id\_pasager

into rez

from pasageri

where upper(n)=upper(nume\_pasager);

DBMS\_OUTPUT.PUT\_LINE('Pasagerul cu numele ' || n || ' si id-ul ' || rez || ', avand id-ul biletului ' || p\_bid(rez) || ' ajunge pe aeroportul unde opereaza compania aeriana ' || p\_can(a));

end p\_pid;

end exercitiul14\_ir;

/

execute exercitiul14\_ir.p\_pid(1000, 'Popa');

execute exercitiul14\_ir.p\_aid(1000);

