Bondoc Ana-Maria

Grupa 241

Proiect SGBD

Gestiunea unei companii aeriene



**CERINTA 1**

Prezentați pe scurt baza de date (utilitatea ei).

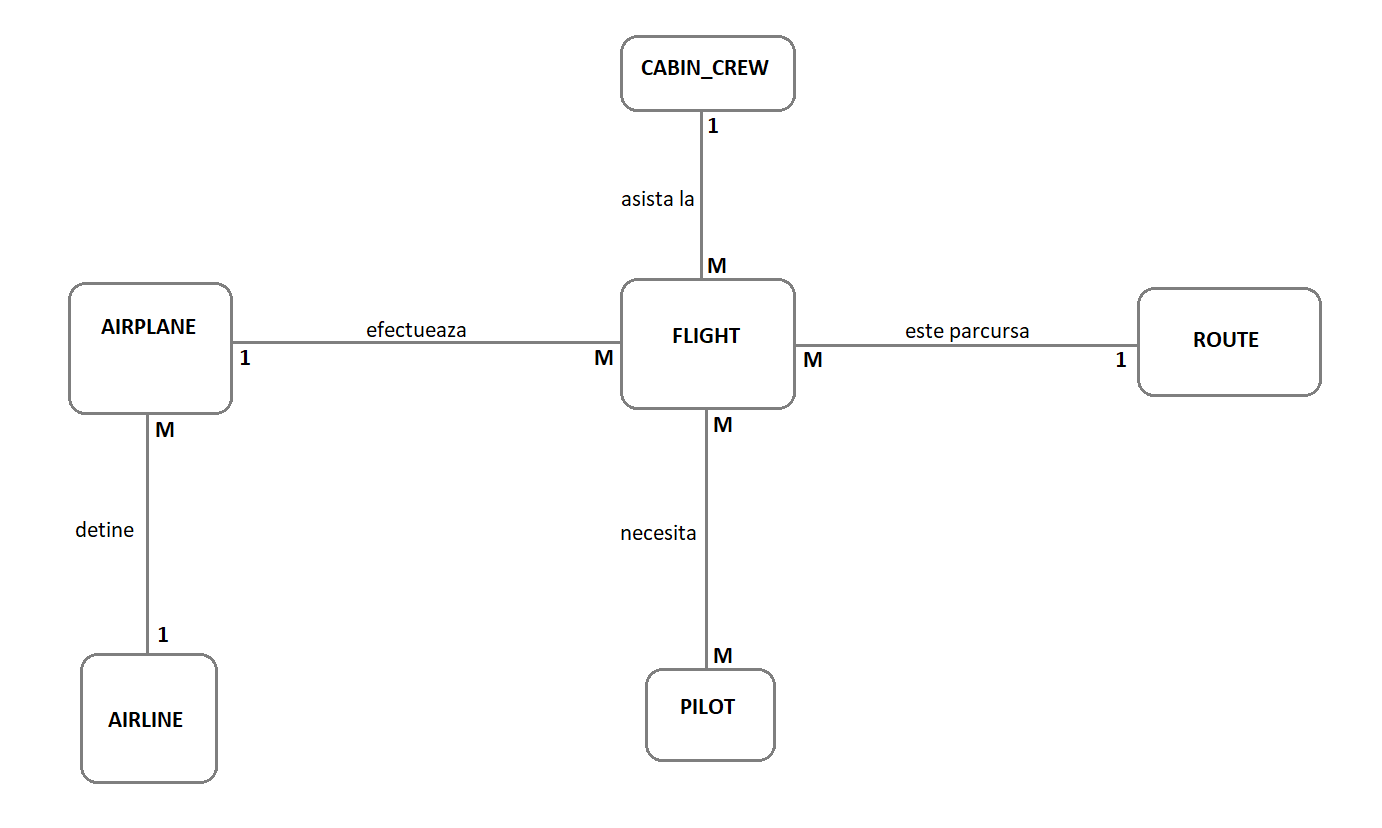
Companiile aeriene mari utilizează numeroase sisteme software pentru diverse scopuri, cum ar fi planificarea flotei, întreținerea aeronavelor, publicațiile tehnice, planificarea zborurilor, programarea echipajului, planificarea personalului, prețurile, gestionarea veniturilor, rezervările de pasageri, rezervarea online, răspunsul vocal interactiv, managementul relațiilor cu clienții, mărfurile, rezervări, autoservire aeroport, urmărire zboruri, navigare, greutate și centrare, geantă electronică de zbor, contabilitate venituri, registru general, management document etc.

Cele mai multe dintre aceste sisteme sunt furnizate de furnizori externi care utilizează diverse baze de date, inclusiv Oracle, SQL Server și MySQL. Bănuiesc că fiecare companie aeriană importantă din SUA folosește cel puțin o instanță din fiecare sistem major de baze de date, găzduit în sediul propriu sau al furnizorilor lor.

Baza de date proiectată în acest proiect asigură buna funcționare a zborurilor efectuate de diverse avioane, aparținând unor companii aeriene. Piloții duc la bun sfârșit zborurile, pe diferite rute, împreună cu echipajul de la bord.

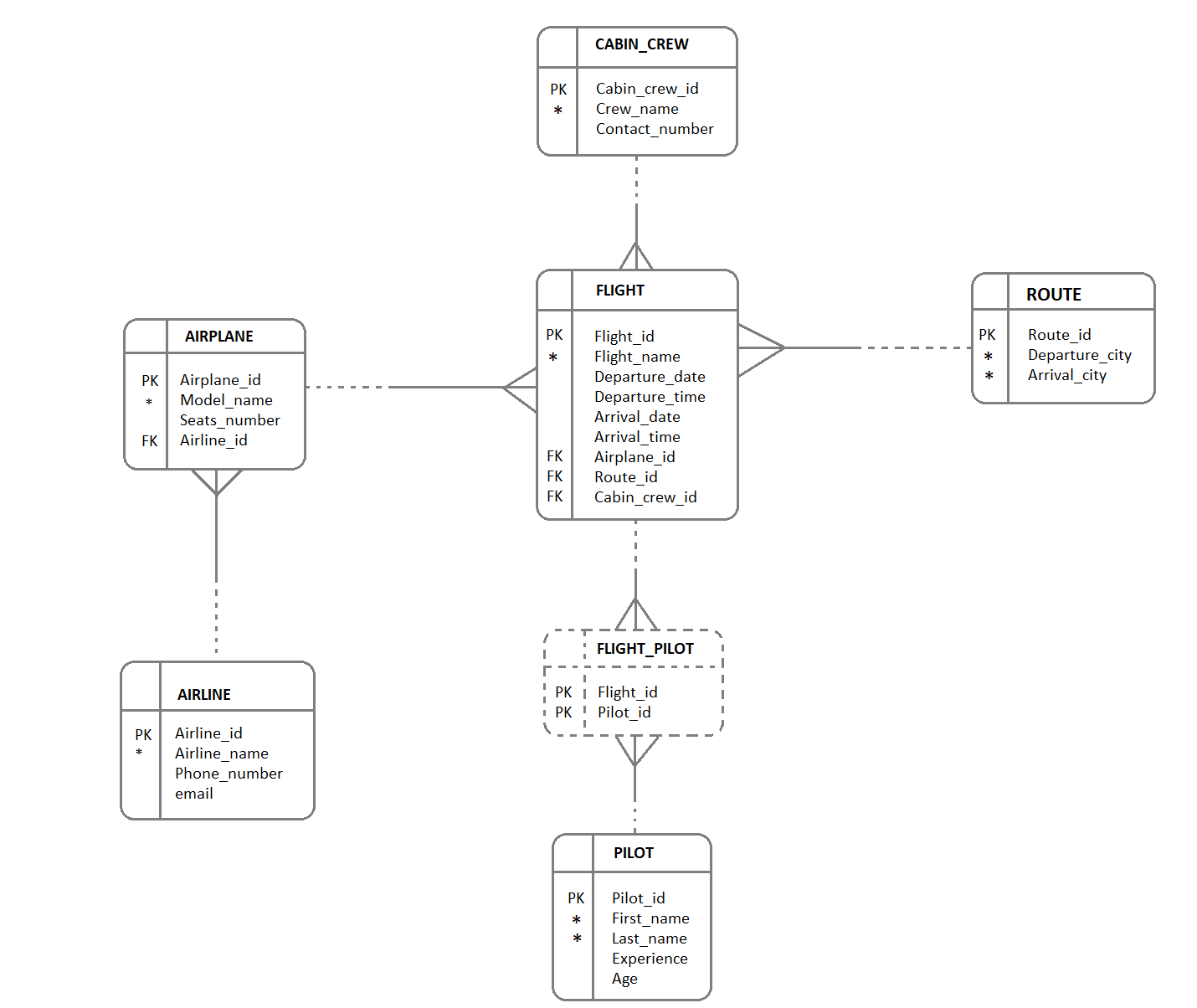
**CERINTA 2**

Realizați diagrama entitate-relație (ERD).

****

**CERINTA 3**

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

****

**CERINTA 4**

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

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

--crearea tabelelor cu cheile primare-------------------------------------------

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

CREATE TABLE airline (

airline\_id NUMBER(9) NOT NULL,

airline\_name VARCHAR2(30) NOT NULL,

phone\_number VARCHAR2(11),

email VARCHAR2(30)

);

ALTER TABLE airline ADD CONSTRAINT airline\_pk PRIMARY KEY ( airline\_id );

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

CREATE TABLE airplane (

airplane\_id NUMBER(9) NOT NULL,

model\_name VARCHAR2(30) NOT NULL,

seats\_number NUMBER(9),

airline\_id NUMBER(9) NOT NULL

);

ALTER TABLE airplane ADD CONSTRAINT airplane\_pk PRIMARY KEY (airplane\_id);

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

CREATE TABLE cabin\_crew (

cabin\_crew\_id NUMBER(9) NOT NULL,

crew\_name VARCHAR2(30) NOT NULL,

contact\_number VARCHAR2(11)

);

ALTER TABLE cabin\_crew ADD CONSTRAINT cabin\_crew\_pk PRIMARY KEY ( cabin\_crew\_id );

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

CREATE TABLE flight (

flight\_id NUMBER(9) NOT NULL,

flight\_name VARCHAR2(30) NOT NULL,

departure\_date DATE,

departure\_time varchar(6),

arrival\_date DATE,

arrival\_time varchar(6),

airplane\_id NUMBER(9) NOT NULL,

route\_id NUMBER(9) NOT NULL,

cabin\_crew\_id NUMBER(9) NOT NULL

);

ALTER TABLE flight

ADD CONSTRAINT flight\_pk PRIMARY KEY ( flight\_id);

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

CREATE TABLE flight\_pilot (

flight\_id NUMBER(9) NOT NULL,

pilot\_id NUMBER(9) NOT NULL );

ALTER TABLE flight\_pilot

ADD CONSTRAINT flight\_pilot\_pk PRIMARY KEY ( flight\_id,

pilot\_id );

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

CREATE TABLE pilot (

pilot\_id NUMBER(9) NOT NULL,

first\_name VARCHAR2(30) NOT NULL,

last\_name VARCHAR2(30) NOT NULL,

experience NUMBER(3),

age NUMBER(3)

);

ALTER TABLE pilot ADD CONSTRAINT pilot\_pk PRIMARY KEY ( pilot\_id );

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

CREATE TABLE route (

route\_id NUMBER(9) NOT NULL,

departure\_city VARCHAR2(30) NOT NULL,

arrival\_city VARCHAR2(30) NOT NULL

);

ALTER TABLE route ADD CONSTRAINT route\_pk PRIMARY KEY ( route\_id );

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

--adaugare foreign key----------------------------------------------------------

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

ALTER TABLE airplane

ADD CONSTRAINT airplane\_airline\_fk FOREIGN KEY ( airline\_id )

REFERENCES airline ( airline\_id);

ALTER TABLE flight

ADD CONSTRAINT flight\_airplane\_fk FOREIGN KEY ( airplane\_id )

REFERENCES airplane ( airplane\_id);

ALTER TABLE flight

ADD CONSTRAINT flight\_cabin\_crew\_fk FOREIGN KEY ( cabin\_crew\_id )

REFERENCES cabin\_crew ( cabin\_crew\_id );

ALTER TABLE flight\_pilot

ADD CONSTRAINT pilot\_flight\_fk FOREIGN KEY ( flight\_id )

REFERENCES flight ( flight\_id );

ALTER TABLE flight\_pilot

ADD CONSTRAINT flight\_pilot\_fk FOREIGN KEY ( pilot\_id )

REFERENCES pilot ( pilot\_id );

ALTER TABLE flight

ADD CONSTRAINT flight\_route\_fk FOREIGN KEY ( route\_id )

REFERENCES route ( route\_id );

**CERINTA 5**

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

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

---inserare date in tabele------------------------------------------------------

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

--AIRLINE-----------------------------------------------------------------------

INSERT INTO airline(airline\_id, airline\_name, phone\_number, email)

VALUES(1, 'WizzAir', '0754178694', 'wizzair@mail.com');

INSERT INTO airline(airline\_id, airline\_name, phone\_number, email)

VALUES(2, 'RyanAir', '0754578914', 'ryanair@mail.com');

INSERT INTO airline(airline\_id, airline\_name, phone\_number, email)

VALUES(3, 'BlueAir', '0710174494', 'blueair@mail.com');

INSERT INTO airline(airline\_id, airline\_name, phone\_number, email)

VALUES(4, 'Lufthansa', '0724475910', 'lufthansa@mail.com');

INSERT INTO airline(airline\_id, airline\_name, phone\_number, email)

VALUES(5, 'Tarom', '0755178451', 'tarom@mail.com');

--AIRPLANE----------------------------------------------------------------------

INSERT INTO airplane(airplane\_id, model\_name, seats\_number, airline\_id)

VALUES(1, 'Airbus A300', 356, 2);

INSERT INTO airplane(airplane\_id, model\_name, seats\_number, airline\_id)

VALUES(2, 'Airbus A321', 412, 4);

INSERT INTO airplane(airplane\_id, model\_name, seats\_number, airline\_id)

VALUES(3, 'Antonov An-26', 589, 4);

INSERT INTO airplane(airplane\_id, model\_name, seats\_number, airline\_id)

VALUES(4, 'Boeing 747', 221, 3);

INSERT INTO airplane(airplane\_id, model\_name, seats\_number, airline\_id)

VALUES(5, 'Boeing 709', 812, 3);

--CABIN\_CREW--------------------------------------------------------------------

INSERT INTO cabin\_crew(cabin\_crew\_id, crew\_name, contact\_number)

VALUES(1, 'Luggage Team', '07234589611');

INSERT INTO cabin\_crew(cabin\_crew\_id, crew\_name, contact\_number)

VALUES(2, 'Cleaning Team', '0723444597');

INSERT INTO cabin\_crew(cabin\_crew\_id, crew\_name, contact\_number)

VALUES(3, 'Medical Team', '0724775558');

INSERT INTO cabin\_crew(cabin\_crew\_id, crew\_name, contact\_number)

VALUES(4, 'Cabin Team', '0755210036');

INSERT INTO cabin\_crew(cabin\_crew\_id, crew\_name, contact\_number)

VALUES(5, 'Emergency Team', '0724774104');

--ROUTE-------------------------------------------------------------------------

INSERT INTO route(route\_id, departure\_city, arrival\_city)

VALUES(1, 'Bucuresti', 'Paris');

INSERT INTO route(route\_id, departure\_city, arrival\_city)

VALUES(2, 'Amsterdam', 'Barcelona');

INSERT INTO route(route\_id, departure\_city, arrival\_city)

VALUES(3, 'Praga', 'Londra');

INSERT INTO route(route\_id, departure\_city, arrival\_city)

VALUES(4, 'Madrid', 'Viena');

INSERT INTO route(route\_id, departure\_city, arrival\_city)

VALUES(5, 'Frankfurt', 'Sibiu');

--PILOT-------------------------------------------------------------------------

INSERT INTO pilot(pilot\_id, first\_name, last\_name, experience, age)

VALUES(1, 'Andrei', 'Dinu', 6, 34);

INSERT INTO pilot(pilot\_id, first\_name, last\_name, experience, age)

VALUES(2, 'Andrei', 'Horceag', 15, 47);

INSERT INTO pilot(pilot\_id, first\_name, last\_name, experience, age)

VALUES(3, 'Mihai', 'Bondoc', 10, 51);

INSERT INTO pilot(pilot\_id, first\_name, last\_name, experience, age)

VALUES(4, 'Adrian', 'Munteanu', 2, 31);

INSERT INTO pilot(pilot\_id, first\_name, last\_name, experience, age)

VALUES(5, 'Razvan', 'Popa', 6, 34);

--FLIGHT------------------------------------------------------------------------

INSERT INTO flight(flight\_id, flight\_name, departure\_date, departure\_time, arrival\_date, arrival\_time,

airplane\_id, route\_id, cabin\_crew\_id)

VALUES(1, 'F2378', '12.12.2022', '12:45', '12.12.2022', '14:00', 2, 4, 2);

INSERT INTO flight(flight\_id, flight\_name, departure\_date, departure\_time, arrival\_date, arrival\_time,

airplane\_id, route\_id, cabin\_crew\_id)

VALUES(2, 'D0122', '31.08.2021', '23:00', '01.09.2021', '03:00', 3, 2, 5);

INSERT INTO flight(flight\_id, flight\_name, departure\_date, departure\_time, arrival\_date, arrival\_time,

airplane\_id, route\_id, cabin\_crew\_id)

VALUES(3, 'G3295', '03.05.2022', '09:30', '03.05.2022', '17:00', 1, 1, 4);

INSERT INTO flight(flight\_id, flight\_name, departure\_date, departure\_time, arrival\_date, arrival\_time,

airplane\_id, route\_id, cabin\_crew\_id)

VALUES(4, 'H4512', '08.01.2023', '17:00', '09.01.2023', '02:00', 5, 3, 1);

INSERT INTO flight(flight\_id, flight\_name, departure\_date, departure\_time, arrival\_date, arrival\_time,

airplane\_id, route\_id, cabin\_crew\_id)

VALUES(5, 'J0065', '03.03.2023', '13:05', '04.03.2023', '13:00', 4, 5, 3);

--FLIGHT\_PILOT------------------------------------------------------------------

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(1, 2);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(1, 4);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(2, 1);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(2, 4);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(3, 1);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(3, 5);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(4, 3);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(4, 2);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(5, 1);

INSERT INTO flight\_pilot(flight\_id, pilot\_id)

VALUES(5, 4);

**CERINTA 6**

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.

--EXERCITIUL 6-------------------------------------------------------------------

-- Definiti un subprogram prin care sa obtinem data si ora decolarii si aterizarii

-- a zborurilor care au ca unul din piloti pe Andrian Munteanu

CREATE OR REPLACE PROCEDURE InformatiiZbor

IS

TYPE zboruri is varray(101) of number;

type ZBOR is record

(flight\_id flight.flight\_id%type,

departure\_date flight.departure\_date%type,

departure\_time flight.departure\_time%type,

arrival\_date flight.arrival\_date%type,

arrival\_time flight.arrival\_time%type);

record ZBOR;

v\_zboruri zboruri;

BEGIN

--selectam zborurile care au ca unul din piloti pe Adrian Munteanu

select distinct

f.flight\_id

bulk collect into

v\_zboruri

from flight\_pilot f

where (select last\_name

from pilot p

where p.pilot\_id=f.pilot\_id) = 'Munteanu'

and (select first\_name

from pilot p

where p.pilot\_id=f.pilot\_id) = 'Adrian';

FOR I IN v\_zboruri.FIRST..v\_zboruri.LAST LOOP

select flight\_id, departure\_date, departure\_time, arrival\_date, arrival\_time

into record

from flight

where flight\_id=v\_zboruri(i);

DBMS\_OUTPUT.PUT\_LINE('Zborul cu id-ul ' || record.flight\_id || ' decoleaza pe data de ' || record.departure\_date ||

' la ora ' || record.departure\_time || ' si ajunge pe data de ' || record.arrival\_date || ' ora ' || record.arrival\_time);

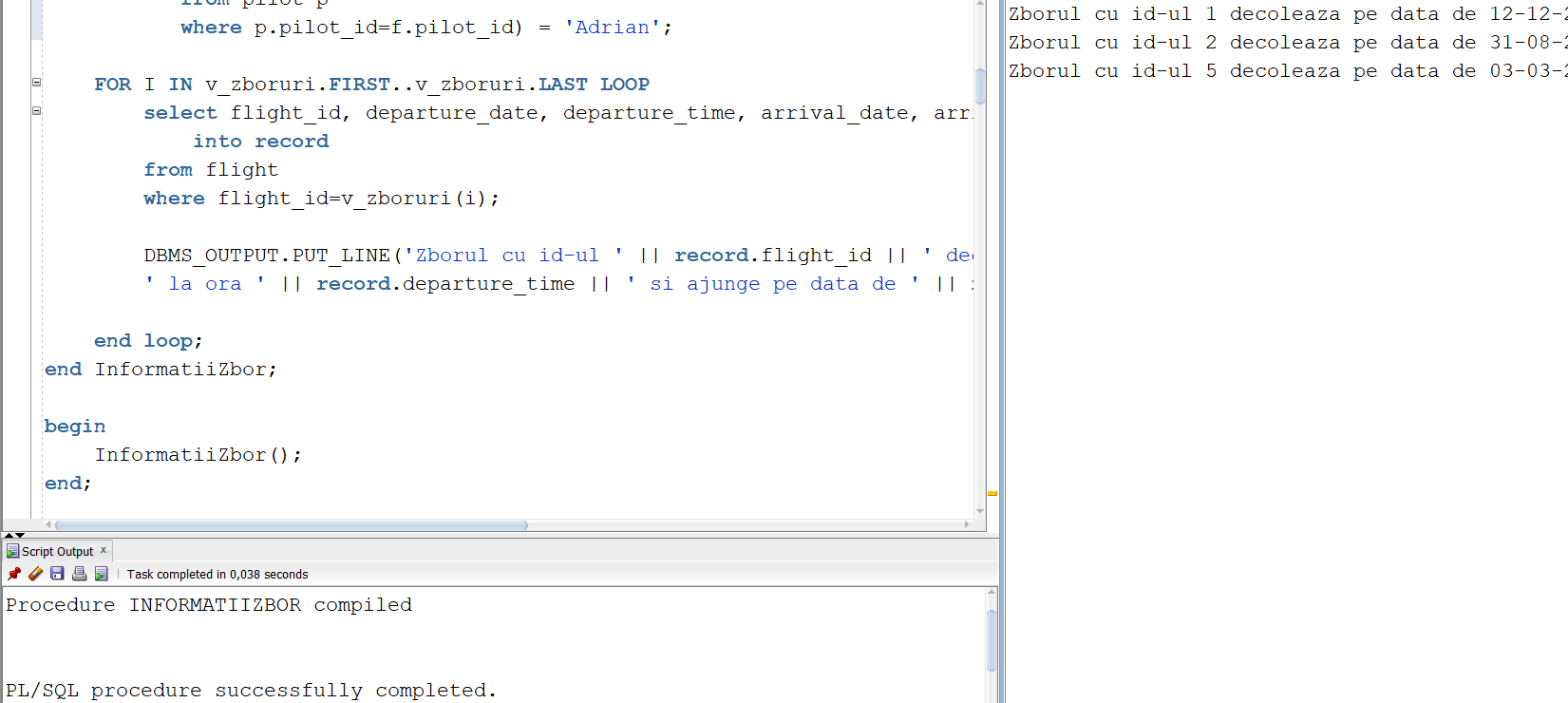
end loop;

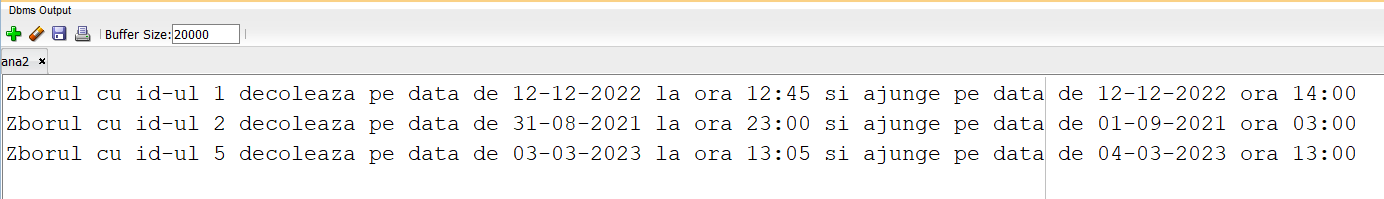
end InformatiiZbor;

begin

InformatiiZbor();

end;





**CERINTA 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. Apelați subprogramul.

--EXERCITIUL 7-------------------------------------------------------------------

-- Obtineti companiile aeriane (numele acesteia si numarul de avioane)

-- care detin cel putin x avioane (x nr introdus de la tastatura)

-- si afiseaza toate avioanele cu zborurile efectuate la momentul actual

CREATE OR REPLACE PROCEDURE CompaniiAeriene

IS

v\_x number(4) := &p\_x;

v\_nr number(4);

v\_nume airline.airline\_name%TYPE;

CURSOR c (parametru NUMBER) IS

SELECT airline\_name nume, COUNT(airplane\_id) nr

FROM airline a, airplane b

WHERE a.airline\_id=b.airline\_id

GROUP BY airline\_name

HAVING COUNT(airplane\_id) > parametru;

v\_nr2 number(4);

v\_nume2 airline.airline\_name%TYPE;

CURSOR c2 IS

SELECT model\_name nume2, COUNT(flight\_id) nr2

FROM airplane a, flight f

WHERE a.airplane\_id=f.airplane\_id

GROUP BY model\_name;

BEGIN

OPEN c(v\_x);

LOOP

FETCH c INTO v\_nume, v\_nr;

EXIT WHEN c%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Compania aeriana '|| v\_nume || ' detine ' || v\_nr || ' avioane');

END LOOP;

CLOSE c;

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

OPEN c2;

LOOP

FETCH c2 INTO v\_nume2, v\_nr2;

EXIT WHEN c2%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Avionul '|| v\_nume2 || ' a efectuat ' || v\_nr2 || ' zboruri');

END LOOP;

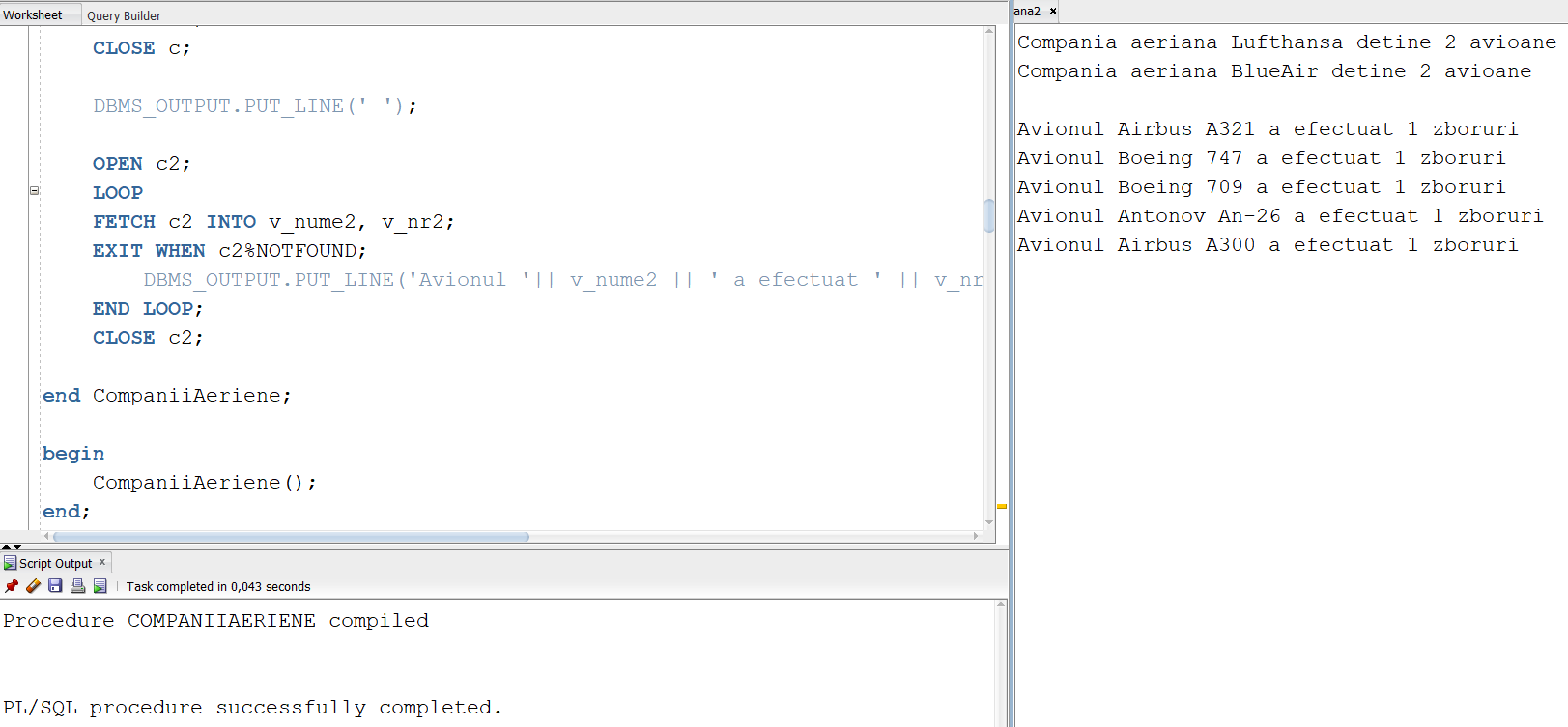
CLOSE c2;

end CompaniiAeriene;

begin

CompaniiAeriene();

end;



**CERINTA 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. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

--EXERCITIUL 8-------------------------------------------------------------------

-- Sa se se afiseze media varstei pilotilor care au dus la

-- bun sfarsit zborul a carui nume este dat de la tastatura

CREATE OR REPLACE FUNCTION functie

(v\_flight\_name flight.flight\_name%TYPE)

RETURN NUMBER IS

varsta pilot.age%type;

data\_decolare flight.departure\_date%type;

BEGIN

--select pentru aruncarea exceptiilor

select departure\_date into data\_decolare from flight

where flight\_name = v\_flight\_name;

SELECT avg(age) into varsta from

pilot p join flight\_pilot fp on p.pilot\_id = fp.pilot\_id

join flight f on fp.flight\_id = f.flight\_id

WHERE flight\_name = v\_flight\_name;

return varsta;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista zboruri cu numele dat');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe zboruri cu numele dat');

WHEN OTHERS THEN

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

END functie;

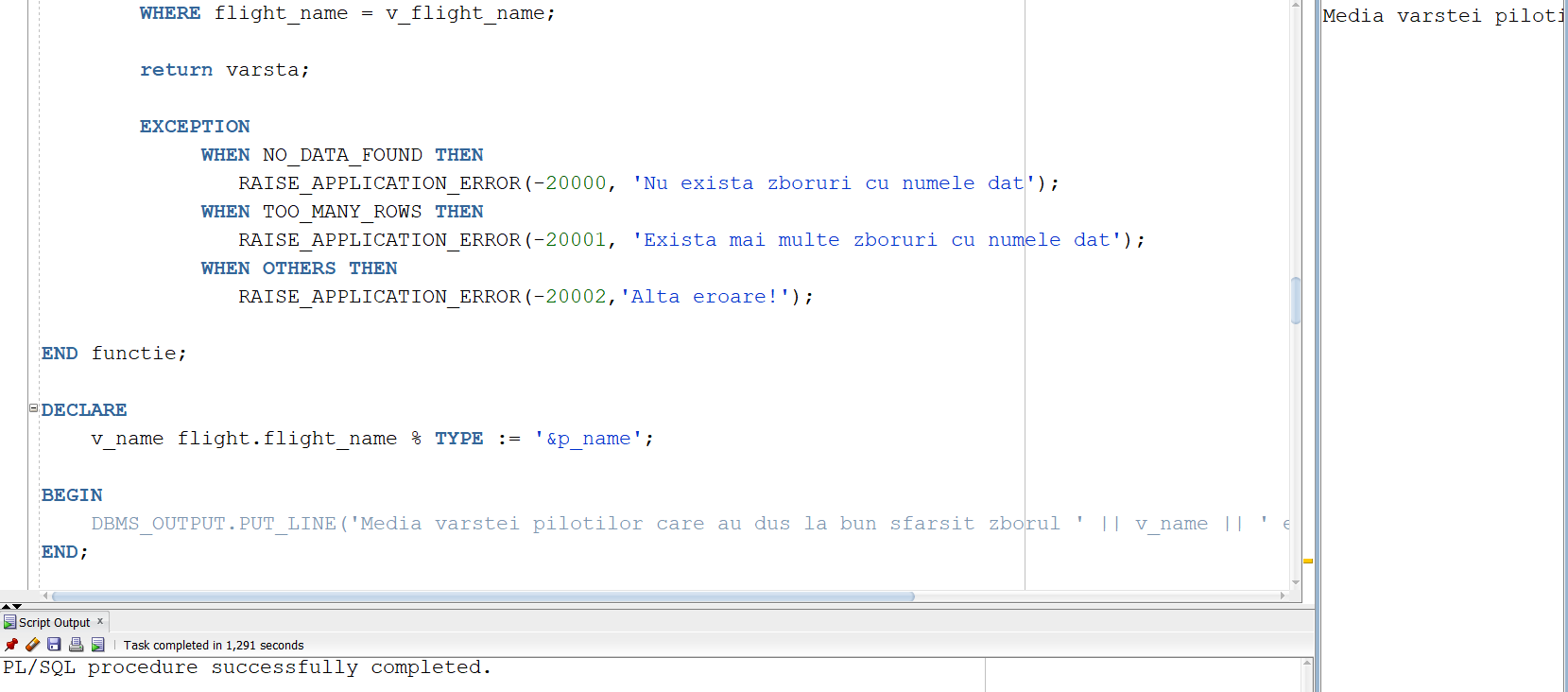
DECLARE

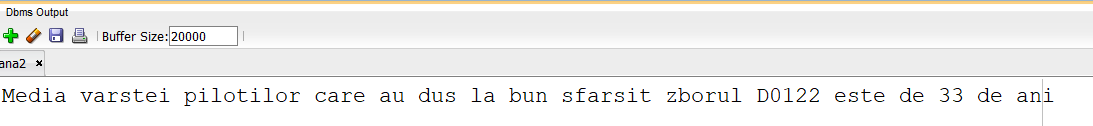
v\_name flight.flight\_name % TYPE := '&p\_name';

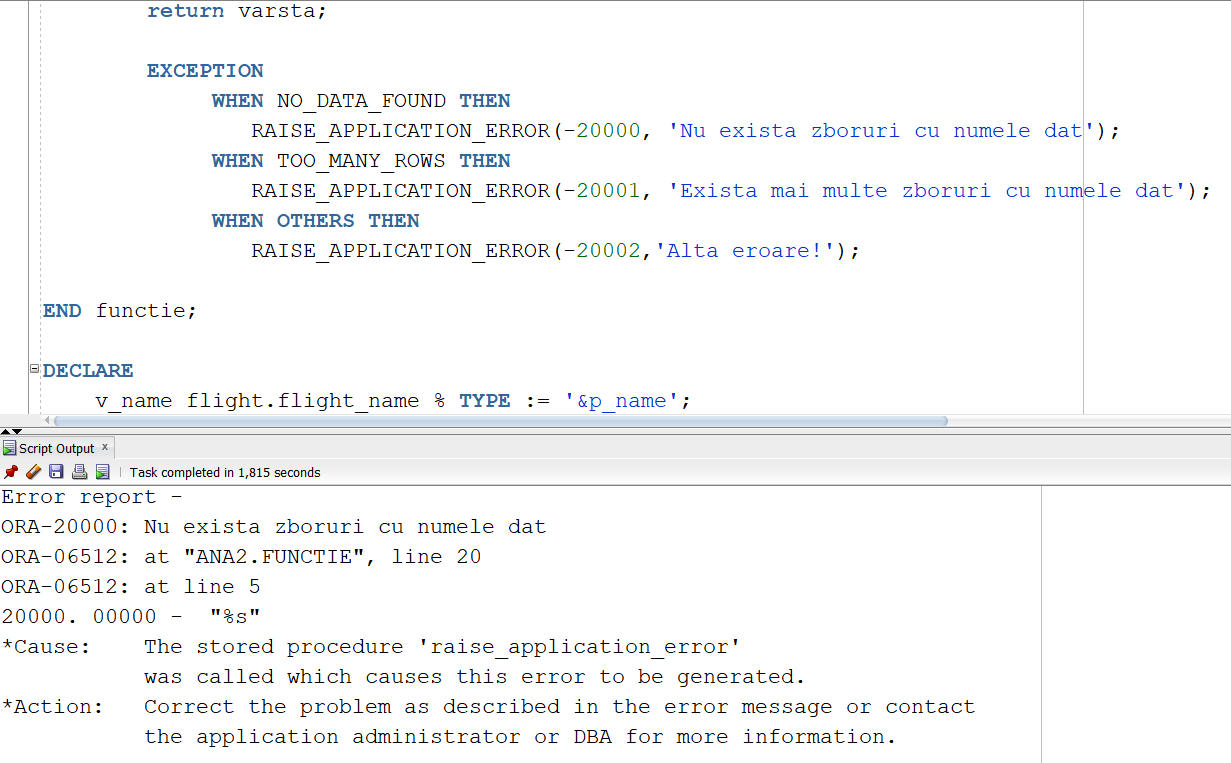
BEGIN

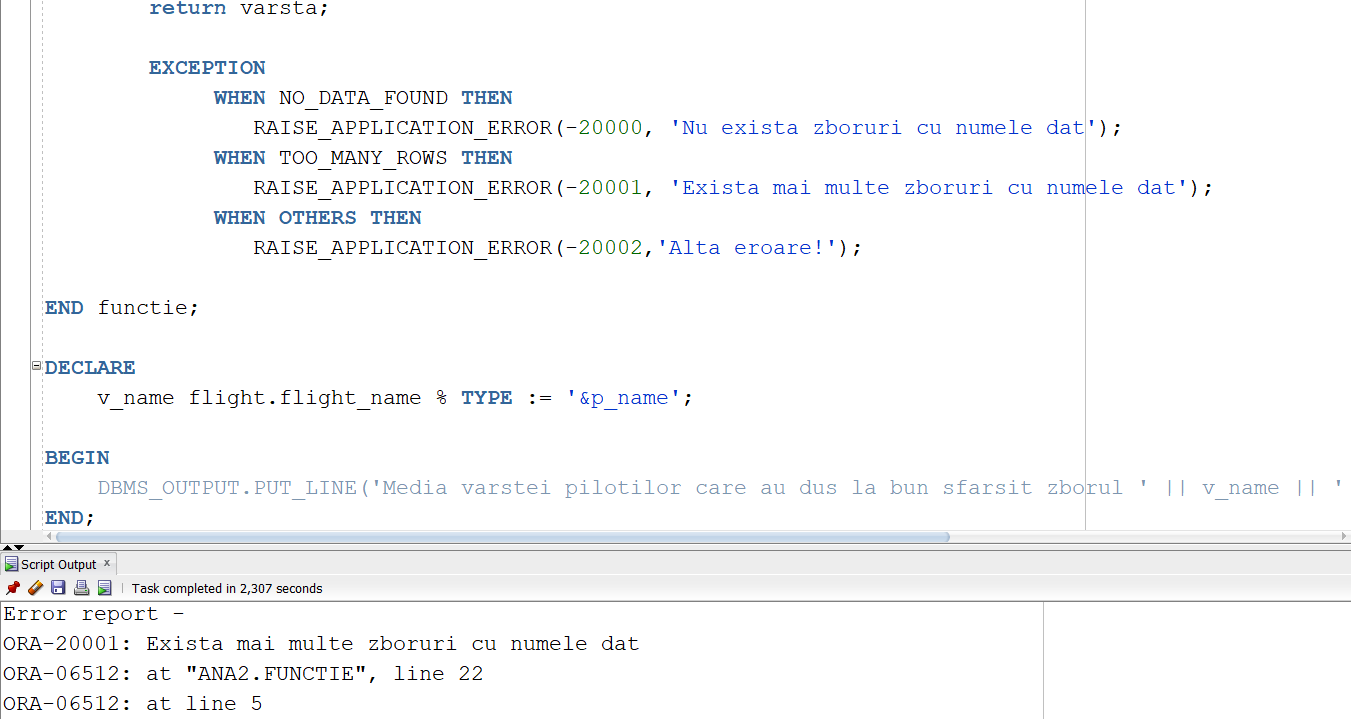
DBMS\_OUTPUT.PUT\_LINE('Media varstei pilotilor care au dus la bun sfarsit zborul ' || v\_name || ' este de ' || functie(v\_name) || ' de ani');

END;









**CERINTA 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.

--EXERCITIUL 9---------------------------------------------------------------------

-- Sa se se afiseze numele, numele pilotului, numele echipei de la bord,

-- si orasul de delocare si aterizare ale unui zbor carui nume este dat

-- de la tastatura

CREATE OR REPLACE PROCEDURE procedura

(v\_flight\_name flight.flight\_name%TYPE)

IS

data\_decolare flight.departure\_date%type;

nume\_zbor flight.flight\_name%type;

prenume\_pilot pilot.first\_name%type;

nume\_pilot pilot.last\_name%type;

nume\_echipaj cabin\_crew.crew\_name%type;

oras\_decolare route.departure\_city%type;

oras\_aterizare route.arrival\_city%type;

BEGIN

--select pentru aruncarea exceptiilor

select departure\_date into data\_decolare from flight

where flight\_name = v\_flight\_name;

SELECT

flight\_name, first\_name, last\_name, crew\_name, departure\_city, arrival\_city

into nume\_zbor, prenume\_pilot, nume\_pilot, nume\_echipaj, oras\_decolare, oras\_aterizare

from flight f

join flight\_pilot fp on f.flight\_id = fp.flight\_id

join pilot p on fp.pilot\_id = p.pilot\_id

join cabin\_crew c on c.cabin\_crew\_id = f.cabin\_crew\_id

join route r on r.route\_id = f.route\_id

WHERE flight\_name = v\_flight\_name;

DBMS\_OUTPUT.PUT\_LINE(nume\_zbor);

DBMS\_OUTPUT.PUT\_LINE('Pilot: ' || nume\_pilot|| ' ' || prenume\_pilot) ;

DBMS\_OUTPUT.PUT\_LINE('Echipaj: ' || nume\_echipaj);

DBMS\_OUTPUT.PUT\_LINE('Decolare: ' || oras\_decolare || ', Aterizare: ' || oras\_aterizare);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista zboruri cu numele dat');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe zboruri cu numele dat');

WHEN OTHERS THEN

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

END procedura;

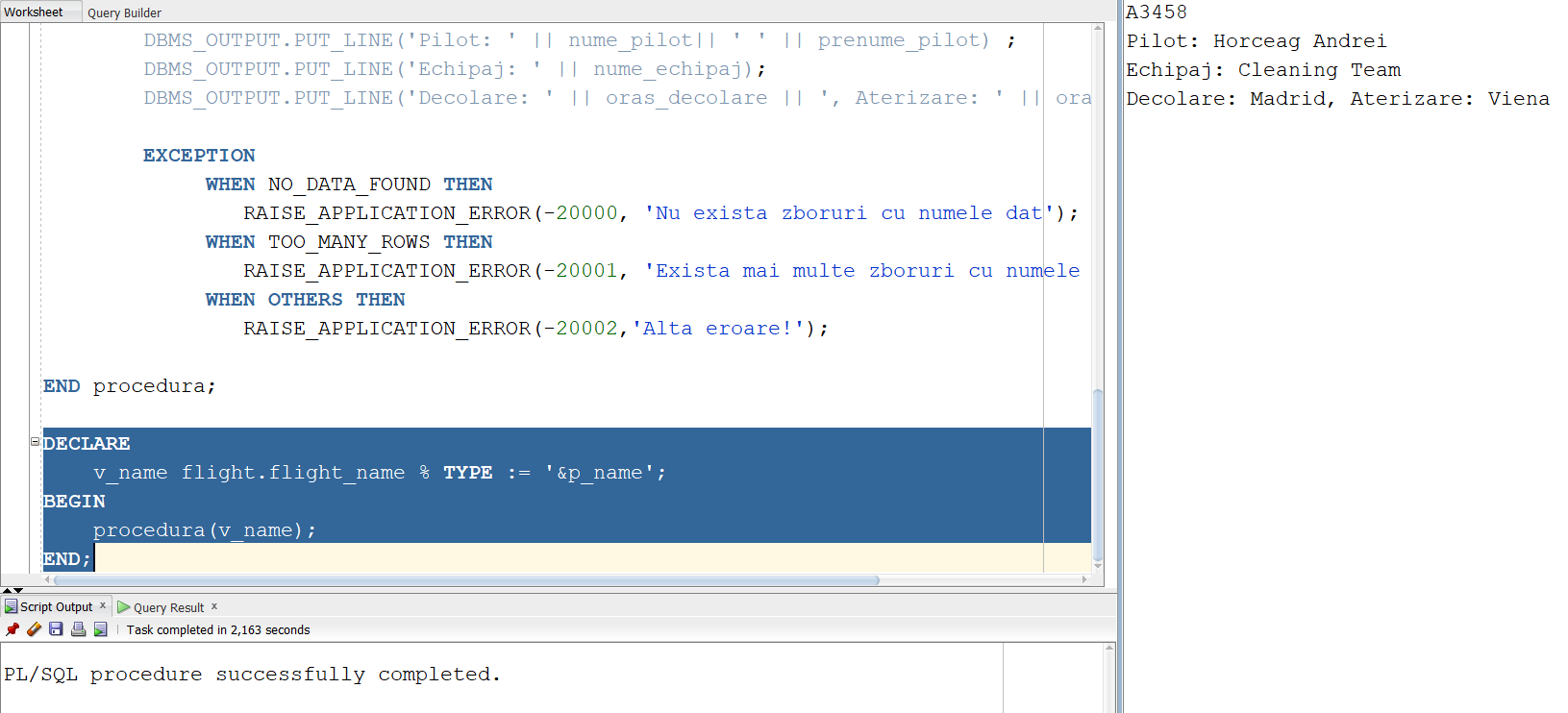
DECLARE

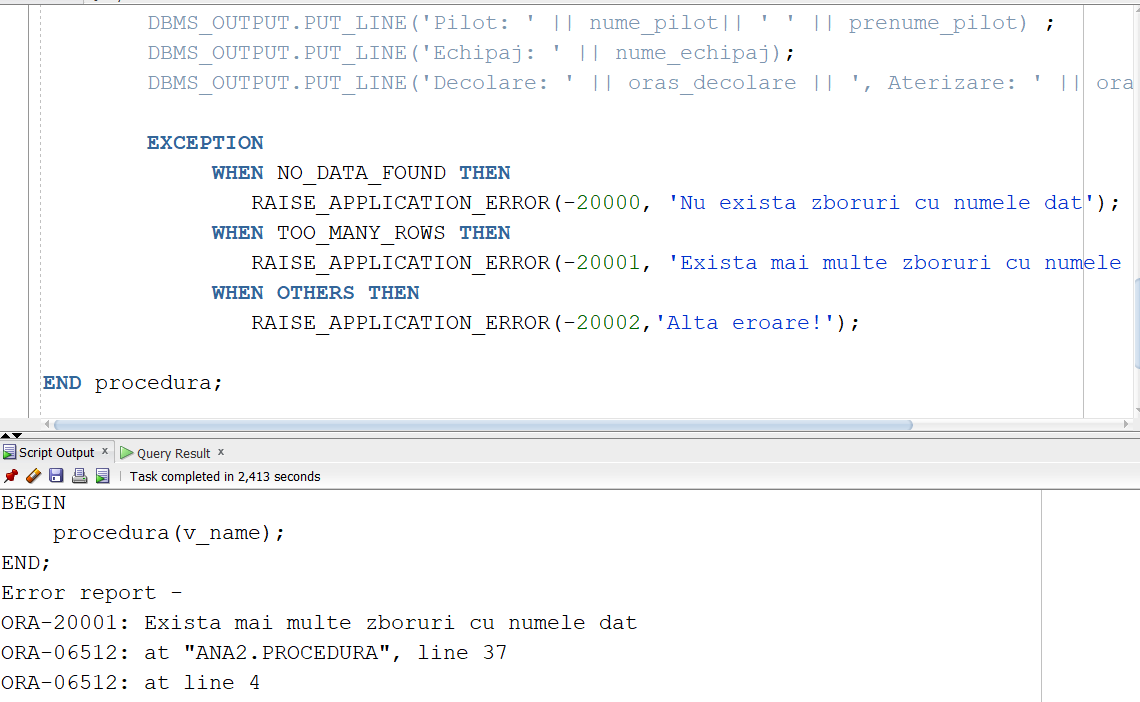
v\_name flight.flight\_name % TYPE := '&p\_name';

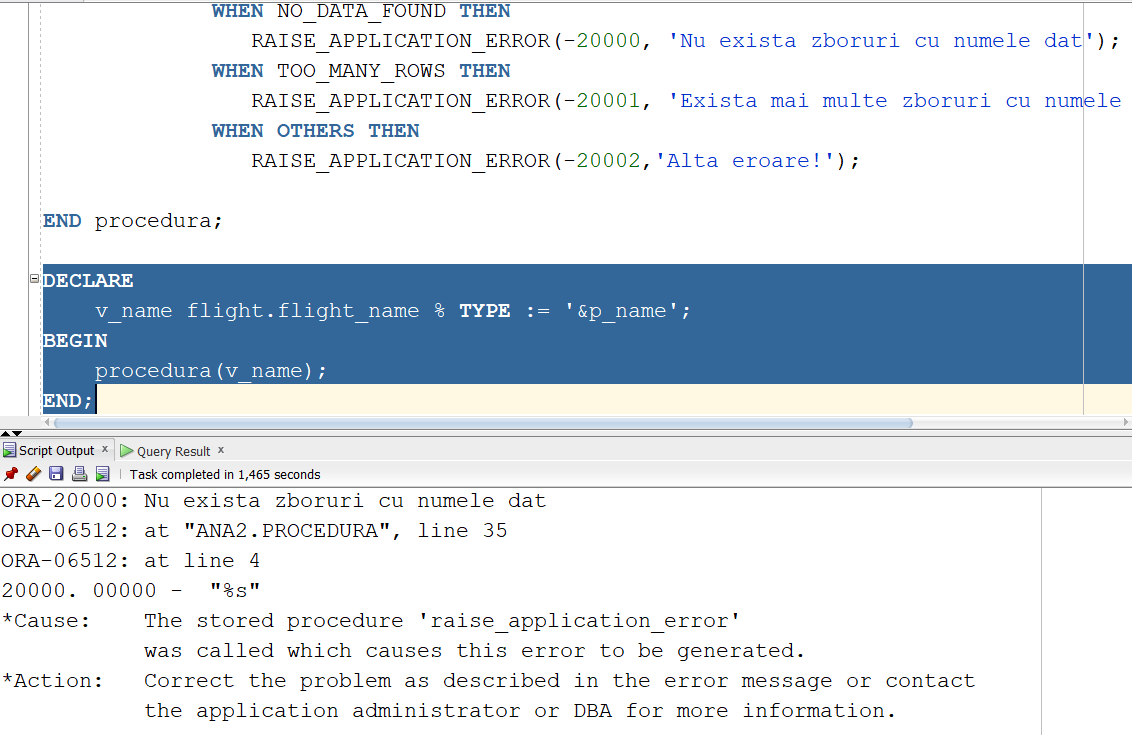
BEGIN

procedura(v\_name);

END;

****

****

****

**CERINTA 10**

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

--EXERCITIUL 10--------------------------------------------------------------------

-- Definiți un declanșator prin care să nu se permită inserarea, stergerea

-- sau actualizarea tabelei route in afara programului de lucru

CREATE OR REPLACE TRIGGER trig

BEFORE INSERT OR DELETE OR UPDATE on route

BEGIN

IF TO\_CHAR(SYSDATE,'HH24') NOT BETWEEN 17 AND 20 THEN

IF INSERTING THEN

RAISE\_APPLICATION\_ERROR(-20001,'Inserarea in tabele este permisa doar in timpul programului de lucru!');

ELSIF DELETING THEN

RAISE\_APPLICATION\_ERROR(-20002,'Stergerea din tabel este permisa doar in timpul programului de lucru!');

ELSE

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

END IF;

END IF;

END;

--INSERARE

INSERT INTO route(route\_id, departure\_city, arrival\_city)

VALUES(6, 'Munchen', 'Napoli');

--STERGERE

DELETE FROM route WHERE departure\_city = 'Bucuresti';

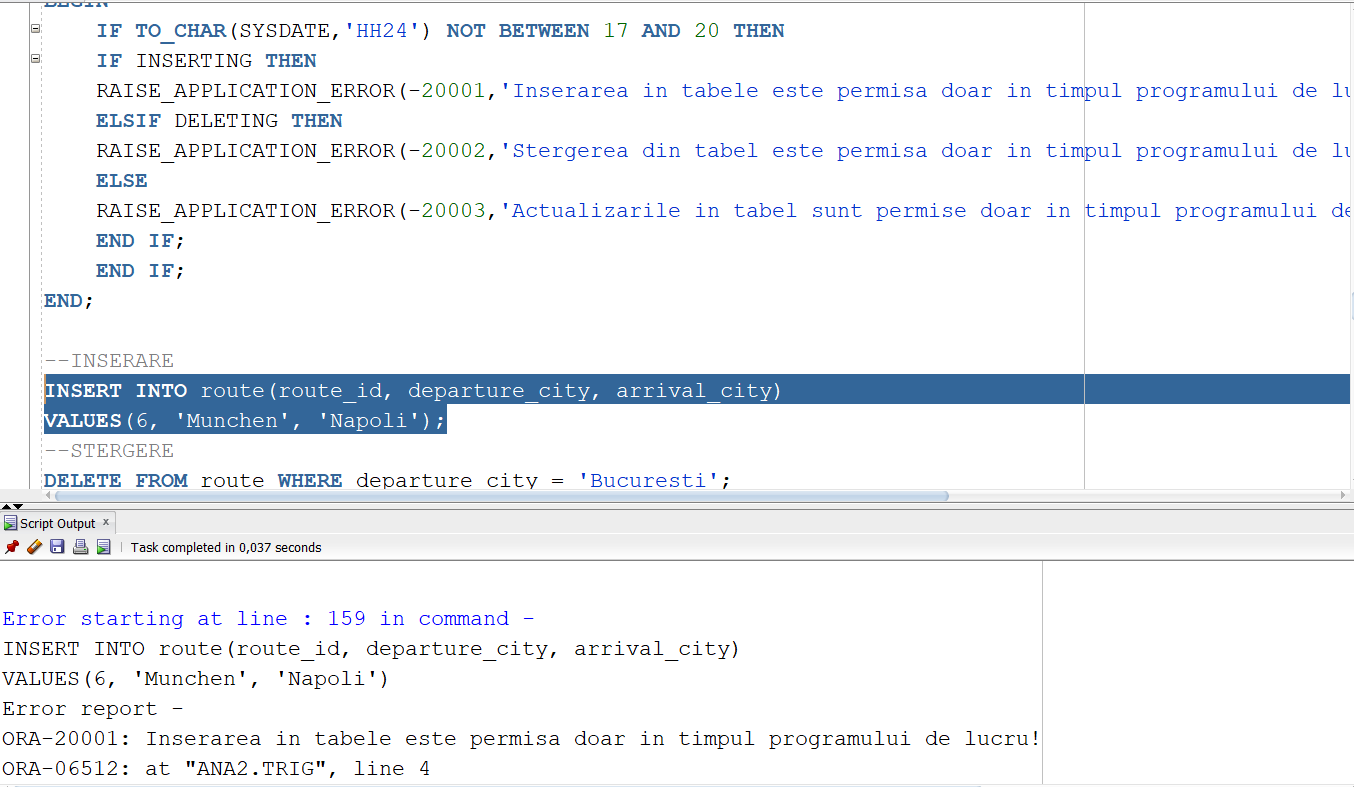
--ACTUALIZARE

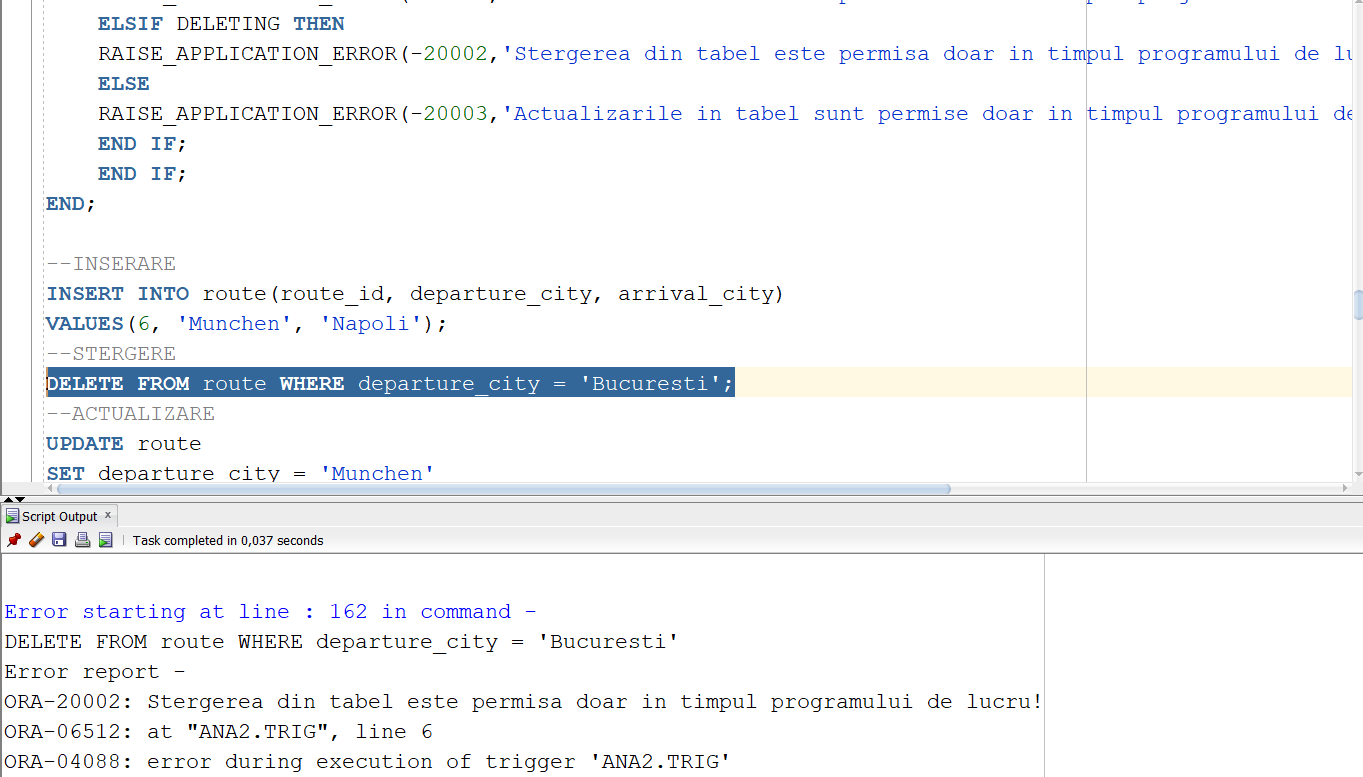
UPDATE route

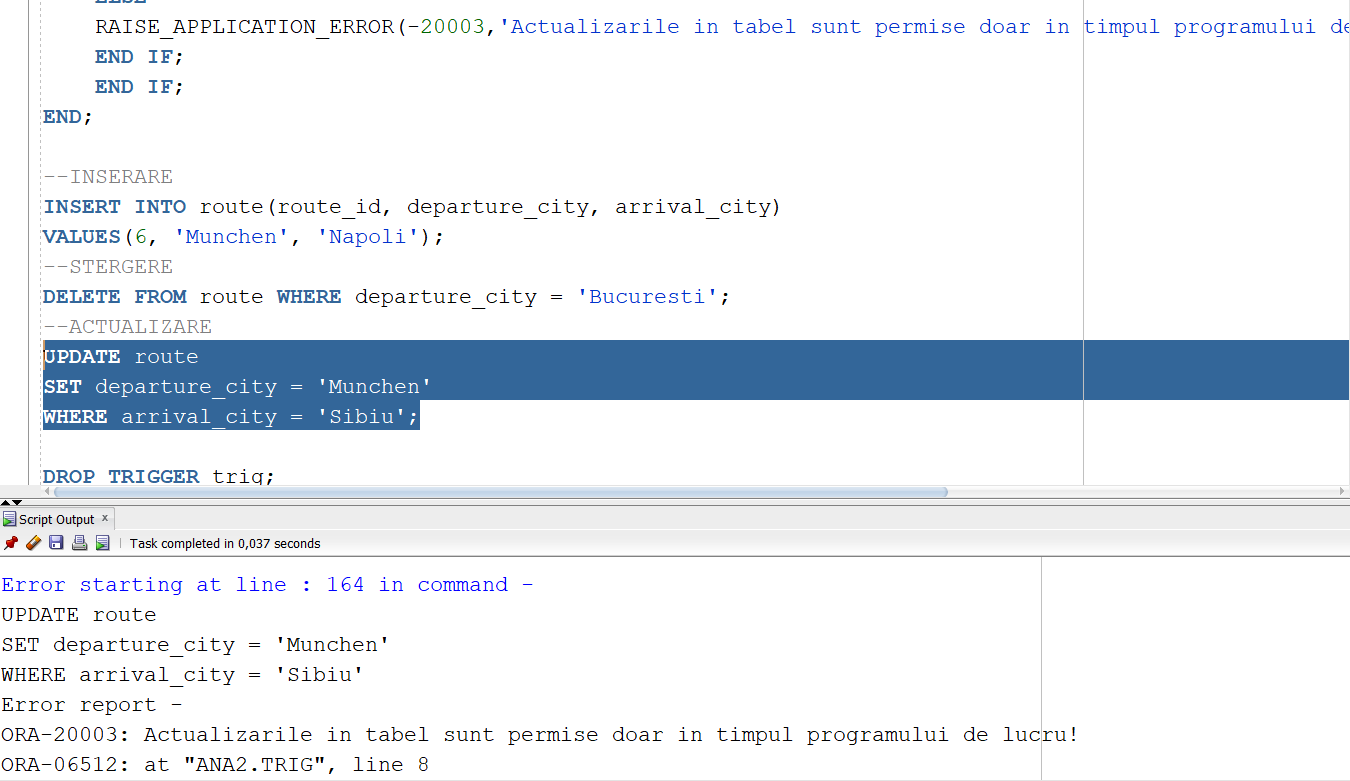
SET departure\_city = 'Munchen'

WHERE arrival\_city = 'Sibiu';

DROP TRIGGER trig;







**CERINTA 11**

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

--EXERCITIUL 11--------------------------------------------------------------------

-- Definiți un declanșator prin care să nu se permită micsorarea sau marirea

-- numarului de locuri din tabela airplane

CREATE OR REPLACE TRIGGER trig2

BEFORE UPDATE OF seats\_number ON airplane

FOR EACH ROW

WHEN (NEW.seats\_number <> OLD.seats\_number)

BEGIN

RAISE\_APPLICATION\_ERROR(-20002,'Numarul de locuri al unui avion nu poate fi modificat');

END;

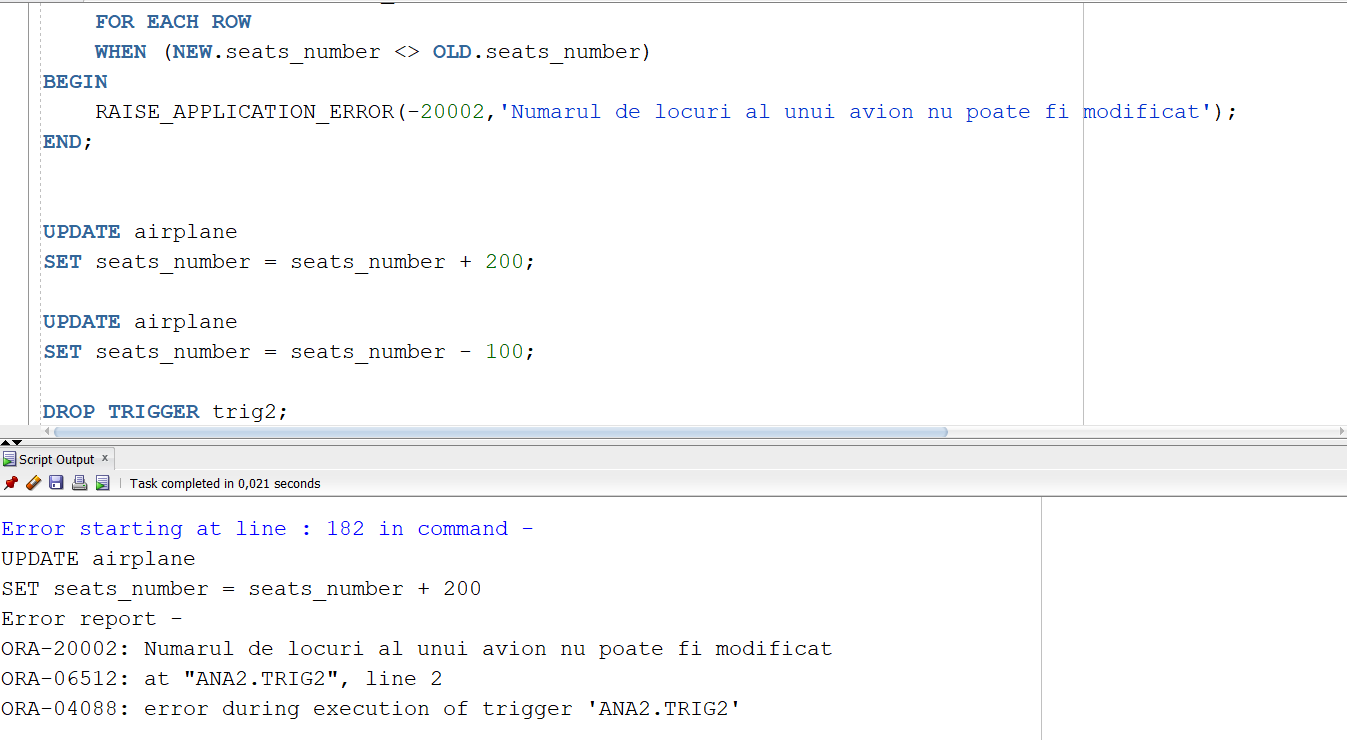
UPDATE airplane

SET seats\_number = seats\_number + 200;

UPDATE airplane

SET seats\_number = seats\_number - 100;

DROP TRIGGER trig2;



**CERINTA 12**

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

--EXERCITIUL 12----------------------------------------------------------------------------

-- Definiți un declanșator care să introducă date în tabela audit\_user după ce

-- utilizatorul a folosit o comandă LDD

CREATE TABLE audit\_user

(

bd\_name VARCHAR2(30),

ddl\_date TIMESTAMP(3),

ddl\_user VARCHAR2(30),

object\_type VARCHAR(30),

object\_name VARCHAR2(30),

ddl\_event VARCHAR2(30)

);

CREATE OR REPLACE TRIGGER trig\_audit

AFTER CREATE OR DROP OR ALTER ON SCHEMA

BEGIN

INSERT INTO audit\_user VALUES(

SYS.DATABASE\_NAME,

SYSTIMESTAMP(3),

SYS.LOGIN\_USER,

SYS.DICTIONARY\_OBJ\_TYPE,

SYS.DICTIONARY\_OBJ\_NAME,

SYS.SYSEVENT

);

END;

--CREATE

CREATE TABLE tabel (coloana\_1 number(2));

--ALTER

ALTER TABLE tabel ADD (coloana\_2 number(2));

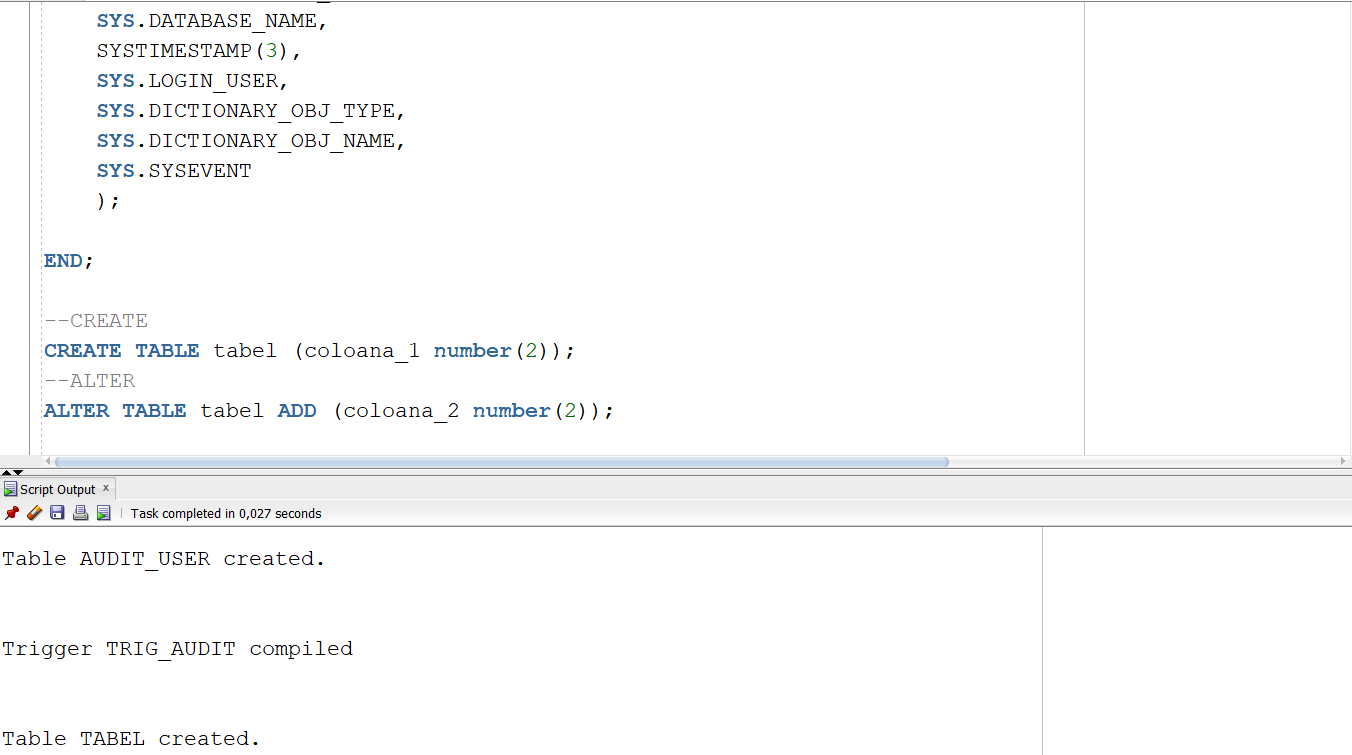
INSERT INTO tabel VALUES (1,2);

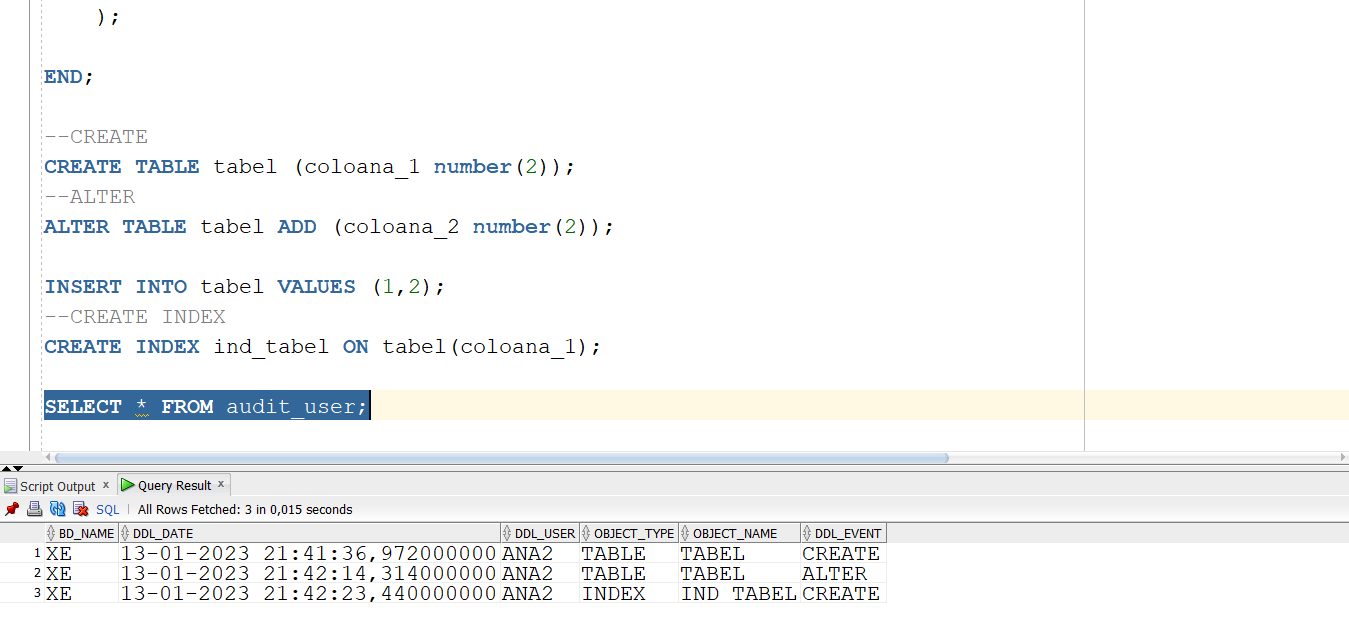
--CREATE INDEX

CREATE INDEX ind\_tabel ON tabel(coloana\_1);

SELECT \* FROM audit\_user;

DROP TRIGGER trig\_audit;





**CERINTA 13**

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

--EXERCITIUL 13-----------------------------------------------------------------

CREATE OR REPLACE PACKAGE pachet as

--6

PROCEDURE InformatiiZbor;

--7

PROCEDURE CompaniiAeriene;

--8

FUNCTION functie(v\_flight\_name flight.flight\_name%TYPE) RETURN NUMBER;

--9

PROCEDURE procedura(v\_flight\_name flight.flight\_name%TYPE);

end pachet;

CREATE OR REPLACE PACKAGE BODY pachet AS

--6

PROCEDURE InformatiiZbor

IS

TYPE zboruri is varray(101) of number;

type ZBOR is record

(flight\_id flight.flight\_id%type,

departure\_date flight.departure\_date%type,

departure\_time flight.departure\_time%type,

arrival\_date flight.arrival\_date%type,

arrival\_time flight.arrival\_time%type);

record ZBOR;

v\_zboruri zboruri;

BEGIN

--selectam zborurile care au ca unul din piloti pe Adrian Munteanu

select distinct

f.flight\_id

bulk collect into

v\_zboruri

from flight\_pilot f

where (select last\_name

from pilot p

where p.pilot\_id=f.pilot\_id) = 'Munteanu'

and (select first\_name

from pilot p

where p.pilot\_id=f.pilot\_id) = 'Adrian';

FOR I IN v\_zboruri.FIRST..v\_zboruri.LAST LOOP

select flight\_id, departure\_date, departure\_time, arrival\_date, arrival\_time

into record

from flight

where flight\_id=v\_zboruri(i);

DBMS\_OUTPUT.PUT\_LINE('Zborul cu id-ul ' || record.flight\_id || ' decoleaza pe data de ' || record.departure\_date ||

' la ora ' || record.departure\_time || ' si ajunge pe data de ' || record.arrival\_date || ' ora ' || record.arrival\_time);

end loop;

end InformatiiZbor;

--7

PROCEDURE CompaniiAeriene

IS

v\_x number(4) := &p\_x;

v\_nr number(4);

v\_nume airline.airline\_name%TYPE;

CURSOR c (parametru NUMBER) IS

SELECT airline\_name nume, COUNT(airplane\_id) nr

FROM airline a, airplane b

WHERE a.airline\_id=b.airline\_id

GROUP BY airline\_name

HAVING COUNT(airplane\_id) > parametru;

v\_nr2 number(4);

v\_nume2 airline.airline\_name%TYPE;

CURSOR c2 IS

SELECT model\_name nume2, COUNT(flight\_id) nr2

FROM airplane a, flight f

WHERE a.airplane\_id=f.airplane\_id

GROUP BY model\_name;

BEGIN

OPEN c(v\_x);

LOOP

FETCH c INTO v\_nume, v\_nr;

EXIT WHEN c%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Compania aeriana '|| v\_nume || ' detine ' || v\_nr || ' avioane');

END LOOP;

CLOSE c;

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

OPEN c2;

LOOP

FETCH c2 INTO v\_nume2, v\_nr2;

EXIT WHEN c2%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Avionul '|| v\_nume2 || ' a efectuat ' || v\_nr2 || ' zboruri');

END LOOP;

CLOSE c2;

end CompaniiAeriene;

--8

FUNCTION functie

(v\_flight\_name flight.flight\_name%TYPE)

RETURN NUMBER IS

varsta pilot.age%type;

data\_decolare flight.departure\_date%type;

BEGIN

--select pentru aruncarea exceptiilor

select departure\_date into data\_decolare from flight

where flight\_name = v\_flight\_name;

SELECT avg(age) into varsta from

pilot p join flight\_pilot fp on p.pilot\_id = fp.pilot\_id

join flight f on fp.flight\_id = f.flight\_id

WHERE flight\_name = v\_flight\_name;

return varsta;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista zboruri cu numele dat');

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe zboruri cu numele dat');

WHEN OTHERS THEN

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

END functie;

--9

PROCEDURE procedura

(v\_flight\_name flight.flight\_name%TYPE)

IS

data\_decolare flight.departure\_date%type;

nume\_zbor flight.flight\_name%type;

prenume\_pilot pilot.first\_name%type;

nume\_pilot pilot.last\_name%type;

nume\_echipaj cabin\_crew.crew\_name%type;

oras\_decolare route.departure\_city%type;

oras\_aterizare route.arrival\_city%type;

BEGIN

--select pentru aruncarea exceptiilor

select departure\_date into data\_decolare from flight

where flight\_name = v\_flight\_name;

SELECT

flight\_name, first\_name, last\_name, crew\_name, departure\_city, arrival\_city

into nume\_zbor, prenume\_pilot, nume\_pilot, nume\_echipaj, oras\_decolare, oras\_aterizare

from flight f

join flight\_pilot fp on f.flight\_id = fp.flight\_id

join pilot p on fp.pilot\_id = p.pilot\_id

join cabin\_crew c on c.cabin\_crew\_id = f.cabin\_crew\_id

join route r on r.route\_id = f.route\_id

WHERE flight\_name = v\_flight\_name;

DBMS\_OUTPUT.PUT\_LINE(nume\_zbor);

DBMS\_OUTPUT.PUT\_LINE('Pilot: ' || nume\_pilot|| ' ' || prenume\_pilot) ;

DBMS\_OUTPUT.PUT\_LINE('Echipaj: ' || nume\_echipaj);

DBMS\_OUTPUT.PUT\_LINE('Decolare: ' || oras\_decolare || ', Aterizare: ' || oras\_aterizare);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu exista zboruri cu numele dat');

WHEN TOO\_MANY\_ROWS THEN

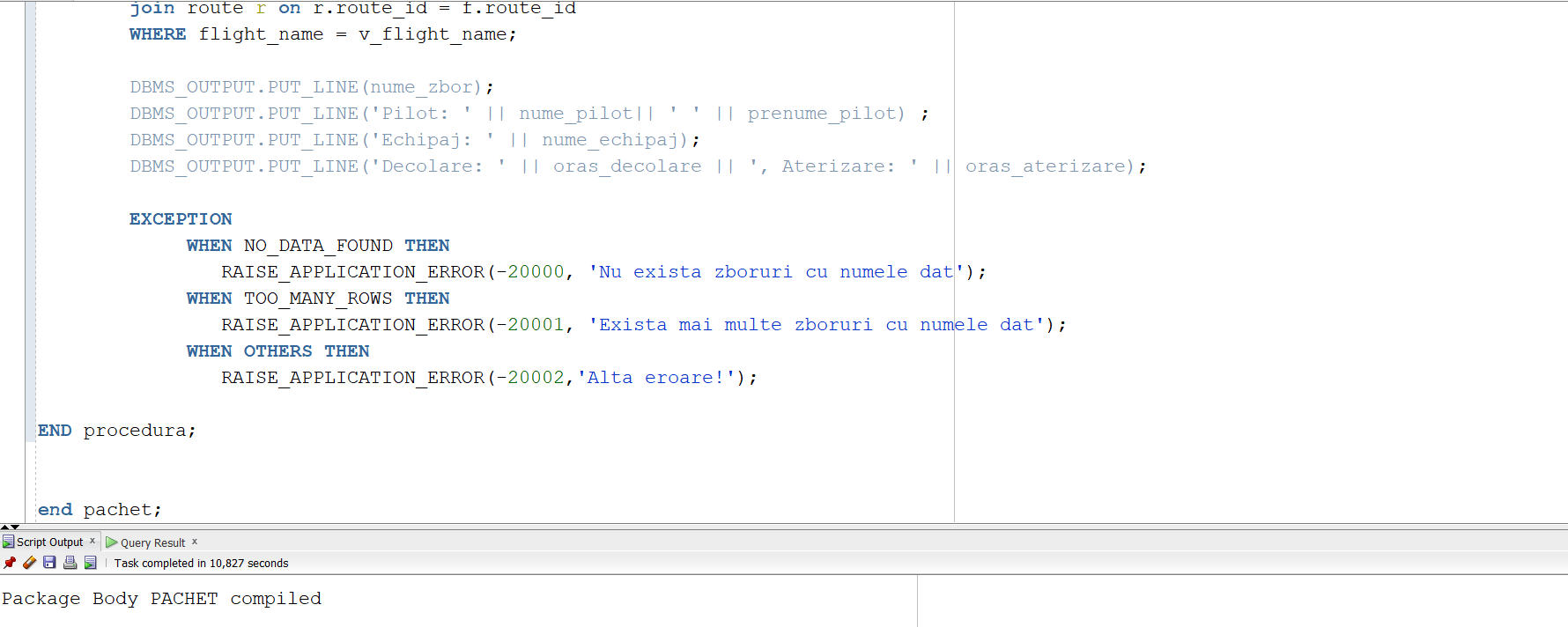
RAISE\_APPLICATION\_ERROR(-20001, 'Exista mai multe zboruri cu numele dat');

WHEN OTHERS THEN

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

END procedura;

end pachet;

****

**CERINTA 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).

--EXERCITIUL 14-----------------------------------------------------------------

CREATE OR REPLACE PACKAGE pachet2 as

--functii

FUNCTION f\_airplane(id\_avion airplane.airplane\_id%TYPE)

RETURN VARCHAR2;

FUNCTION f\_airline(nume\_companie airline.airline\_name%TYPE)

RETURN NUMBER;

--proceduri

PROCEDURE p3(nume\_pilot pilot.last\_name%TYPE, prenume\_pilot pilot.first\_name%TYPE);

PROCEDURE InformatiiRuta;

end pachet2;

CREATE OR REPLACE PACKAGE BODY pachet2 AS

--functii

FUNCTION f\_airplane(id\_avion airplane.airplane\_id%TYPE)

RETURN VARCHAR2 is nume VARCHAR2(30);

BEGIN

SELECT model\_name INTO nume

FROM airplane

WHERE id\_avion = airplane\_id;

return nume;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Date introduse gresit!');

return null;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001,'Alta eroare!');

return null;

END f\_airplane;

FUNCTION f\_airline(nume\_companie airline.airline\_name%TYPE)

RETURN NUMBER is numar NUMBER;

BEGIN

SELECT airline\_id INTO numar

FROM airline

WHERE nume\_companie = airline\_name;

return numar;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Date introduse gresit!');

return null;

WHEN TOO\_MANY\_ROWS THEN

RAISE\_APPLICATION\_ERROR(-20000,'Exista mai multe companii aeriene cu numele dat');

return null;

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001,'Alta eroare!');

return null;

END f\_airline;

--proceduri

PROCEDURE p3(nume\_pilot pilot.last\_name%TYPE, prenume\_pilot pilot.first\_name%TYPE)

IS experienta pilot.experience%TYPE;

BEGIN

SELECT experience INTO experienta

FROM pilot

WHERE last\_name = nume\_pilot and first\_name = prenume\_pilot;

DBMS\_OUTPUT.PUT\_LINE('Experienta de ' || experienta || ' ani');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista pilot cu numele dat');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Exista mai multi piloti cu numele dat');

WHEN OTHERS THEN

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

END p3;

PROCEDURE InformatiiRuta

IS

TYPE rute is varray(101) of number;

type RUTA is record

(oras\_decolare route.departure\_city%type,

oras\_aterizare route.arrival\_city%type);

record RUTA;

v\_rute rute;

id\_ruta route.route\_id%type;

BEGIN

--selectez id-ul rutelor care are unul din orase Bucuresti

SELECT DISTINCT

route\_id

bulk collect into

v\_rute

FROM route

WHERE departure\_city = 'Bucuresti' or arrival\_city = 'Bucuresti';

FOR I IN v\_rute.FIRST..v\_rute.LAST LOOP

select departure\_city, arrival\_city

into record

from route

where route\_id=v\_rute(i);

DBMS\_OUTPUT.PUT\_LINE('Ruta: ' || record.oras\_decolare || ' - ' || record.oras\_aterizare);

end loop;

end InformatiiRuta;

end pachet2;

