Academia de Studii Economice
Facultatea de Cibernetica, Statistica si Informatica Economica

Proiect SGBD Oracle
Gestiunea datelor unei Linii Aeriene

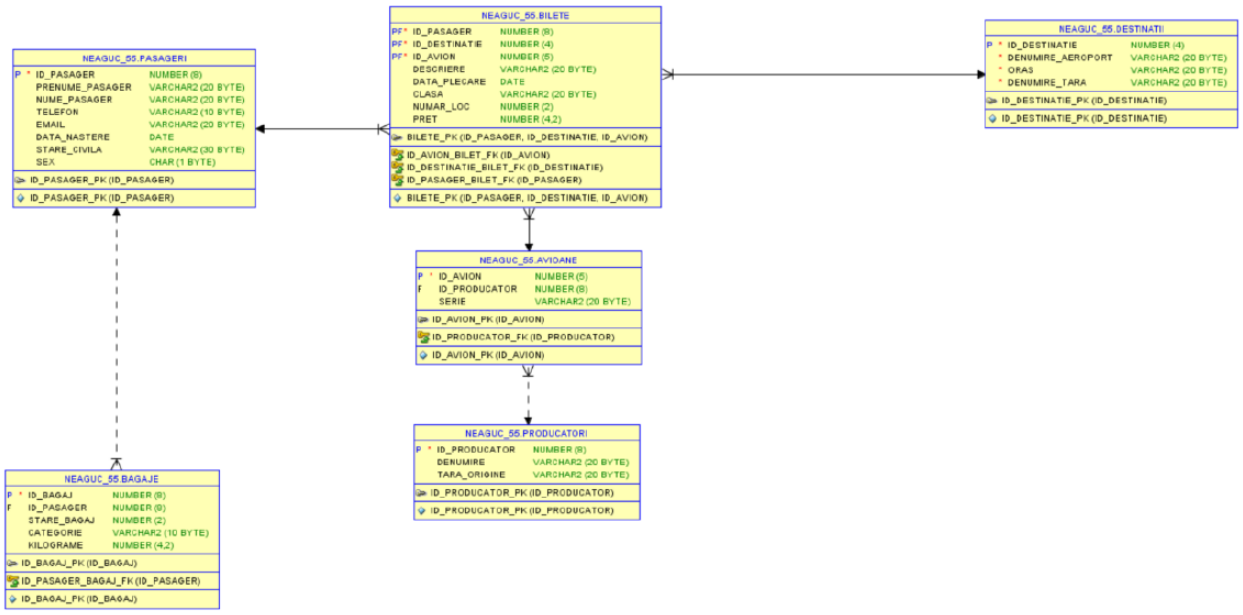
Profesor coordonator
Florea Alexandra-Maria

Student
Neagu Chris-Marios
Grupa 1055D

Cuprins

Descrierea temei

Am construit o baza de date pentru o linie aeriana. Compania efectueaza transport aerian, iar pasagerii pot alege sa calatoreasca la clasa I, business sau econom. In functie de clasa si de dimensiunea bagajului (daca categoria este de cala si bagajul depaseste 10kg spre exemplu se va plati in plus, iar bagajele de mana trebuie sa fie de maxim 10kg pentru a nu ingreuna avionul) se va stabili valoarea finala a biletului. Astfel, se asigura buna gestiune a calatoriilor, cat si bunastarea pasagerilor aflati la bordul avionului.



Comenzi LDD: CREATE;ALTER; DROP

```
SET SERVEROUTPUT ON
```

-- Sa se creeze tabele de backup pentru schema data.

```
BEGIN
EXECUTE IMMEDIATE 'CREATE TABLE PASAGERI_BKP AS SELECT * FROM
PASAGERI';
EXECUTE IMMEDIATE 'CREATE TABLE BILETE_BKP AS SELECT * FROM
BILETE';
EXECUTE IMMEDIATE 'CREATE TABLE DESTINATII_BKP AS SELECT * FROM
DESTINATII';
EXECUTE IMMEDIATE 'CREATE TABLE AVIOANE_BKP AS SELECT * FROM
AVIOANE';
EXECUTE IMMEDIATE 'CREATE TABLE PRODUCATORI_BKP AS SELECT *
FROM PRODUCATORI';
EXECUTE IMMEDIATE 'CREATE TABLE BAGAJE_BKP AS SELECT * FROM
BAGAJE';
END;
/
```

-- Sa se adauge o coloana noua in tabelul producatori bkp

```
DECLARE
V_SIR VARCHAR2(200);
BEGIN
V_SIR:='ALTER TABLE PRODUCATORI_BKP ADD (CIFRA_AFACERI
NUMBER(8,4))';
DBMS_OUTPUT.PUT_LINE(V_SIR);
EXECUTE IMMEDIATE V_SIR;
END;
/
```

-- Sa se stearga toate tabelele create.

```
BEGIN
EXECUTE IMMEDIATE 'DROP TABLE PASAGERI_BKP';
EXECUTE IMMEDIATE 'DROP TABLE BILETE_BKP';
EXECUTE IMMEDIATE 'DROP TABLE DESTINATII_BKP';
EXECUTE IMMEDIATE 'DROP TABLE AVIOANE_BKP';
EXECUTE IMMEDIATE 'DROP TABLE PRODUCATORI_BKP';
EXECUTE IMMEDIATE 'DROP TABLE BAGAJE_BKP';
END;
/
```

Comenzi LMD: INSERT; UPDATE; DELETE

-- Sa se adauge o noua inregistrare in tabela PASAGERI_BKP

```
BEGIN
INSERT INTO PASAGERI_BKP
VALUES (1,'TOMA','AISHA-
ROXANA','0744552645','toma.aisha@gmail.com',TO_DATE('13-08-1999','DD-MM-
YYYY'),'singur','F');
COMMIT;
END;
/
```

-- Sa se reduca pretul biletelor la clasa Econom

```
BEGIN
UPDATE BILETE_BKP SET PRET=PRET*0.95
WHERE CLASA='Econom';
COMMIT;
END;
/
```

-- Sa se stearga angajatul cu numele NEAGU din tabela

```
PASAGERI_BKP
BEGIN
DELETE FROM PASAGERI_BKP WHERE UPPER(NUME_PASAGER) LIKE 'NEA%';
```

```
ROLLBACK;
END;
/
```

Structuri de control

- **IF CU VARIABILA DE SUBSTITUTIE**

-- SA SE AFISEZE DACA EXISTA RESTRICTII DE CIRCULATIE PENTRU REGATUL UNIT

```
DECLARE
BEGIN
IF '&DENUMIRE_TARA' = 'REGATUL UNIT' THEN
DBMS_OUTPUT.PUT_LINE('Nu exista restrictii de circulatie catre aceasta tara in momentul actual, '||SYSDATE);
END IF;
END;
/
```

-- Sa se afiseze starea bagajelor in format codificat si decodificat pentru un bagaj cu un id dat

```
DECLARE
STARE BAGAJE_BKP.STARE_BAGAJ%TYPE;
V_SIR VARCHAR(20);
BEGIN
SELECT STARE_BAGAJ INTO STARE FROM BAGAJE_BKP
WHERE ID_BAGAJ=&ID_BAGAJ;
```

```
CASE
WHEN STARE=0 THEN
   V_SIR:='PRELUAT';
WHEN STARE=1 THEN
   V_SIR:='IN AVION';
ELSE
   V_SIR:='RETURNAT';
END CASE;
DBMS_OUTPUT.PUT_LINE('COD: #'||STARE||' DECODIFICARE: '||V_SIR);
END;
/
```

-- Se afiseaza in ordine pasagerii cu codurile in intervalul 1-3

```
DECLARE
V_NUME PASAGERI_BKP.NUME_PASAGER%TYPE;
V_TELEFON PASAGERI_BKP.TELEFON%TYPE;
I NUMBER(3):=1;
BEGIN
LOOP
SELECT NUME_PASAGER INTO V_NUME FROM PASAGERI_BKP WHERE
ID_PASAGER=I;
SELECT TELEFON INTO V_TELEFON FROM PASAGERI_BKP WHERE
ID_PASAGER=I;
DBMS_OUTPUT.PUT_LINE('NUME PASAGER: '||V_NUME||' TELEFON:
'||V_TELEFON);
I:=I+1;
EXIT WHEN I>3;
END LOOP;
END;
/
```

-- SA SE AFISEZE TOTI PRODUCATORI DIN INTERVALUL 1-2 ATAT TIMP CAT TARA DE ORIGINE NU ESTE RUSIA

```
DECLARE
V_DENUMIRE PRODUCATORI_BKP.DENUMIRE%TYPE;
V_TARA_ORIGINE PRODUCATORI_BKP.TARA_ORIGINE%TYPE;
BEGIN
FOR I IN 1..2 LOOP
SELECT DENUMIRE INTO V_DENUMIRE FROM PRODUCATORI_BKP WHERE
ID_PRODUCATOR=I;
SELECT TARA_ORIGINE INTO V_TARA_ORIGINE FROM PRODUCATORI_BKP
WHERE ID_PRODUCATOR=I;
EXIT WHEN V_TARA_ORIGINE='RUSIA';
DBMS_OUTPUT.PUT_LINE('DENUMIRE: '||V_DENUMIRE||' TARA ORIGINE:
'||V_TARA_ORIGINE);
END LOOP;
END;
/
```

Cursori Impliciti

-- Se sterge din tabela DESTINATII, destinatia a carei ID este introdusa de catre utilizator printr-o variabila de substitutie

```
ACCEPT G_DID PROMPT 'INTRODUCETI ID-UL DESTINATIEI'
VARIABLE MESAJ VARCHAR2(100);
DECLARE
BEGIN
DELETE FROM DESTINATII_BKP WHERE ID_DESTINATIE=&G_DID;
:MESAJ:=TO_CHAR(SQL%ROWCOUNT)||'INREGISTRARI STERSE';
END;
/
PRINT MESAJ;
ROLLBACK;
```

-- SA SE STEARGA TOATE BILETELE CU DATA DE PLECARE INAINTE DE DATA CURENTA

```
BEGIN
DELETE FROM BILETE_BKP
WHERE TRUNC(DATA_PLECARE) < TRUNC(SYSDATE);
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' RANDURI STERSE');
ROLLBACK;
END;
/
```

--SA SE MODIFICE STAREA TUTUROR BAGAJELOR PRELUATE (1) IN BAGAJE CU STAREA RETURNATE (2)

```
BEGIN
UPDATE BAGAJE_BKP
SET STARE_BAGAJ = 2 WHERE STARE_BAGAJ = 1;
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||' RANDURI ACTUALIZATE');
ROLLBACK;
END;
/
```

Cursori Expliciti

-- Sa se afiseze lista cu pasageri a caror stare civila este singur folosing un cursor si 3 variabile

```
DECLARE
V_NUME PASAGERI_BKP.NUME_PASAGER%TYPE;
V_PRENUME PASAGERI_BKP.PRENUME_PASAGER%TYPE;
V_TELEFON PASAGERI_BKP.TELEFON%TYPE;
CURSOR C_PASAGER IS
SELECT NUME_PASAGER, PRENUME_PASAGER, TELEFON
FROM PASAGERI_BKP WHERE STARE_CIVILA = 'singur';
BEGIN
OPEN C_PASAGER;
LOOP
FETCH C_PASAGER INTO V_NUME,V_PRENUME,V_TELEFON;
EXIT WHEN C_PASAGER%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('NUME: '||V_NUME||' PRENUME: '||V_PRENUME||'
TELEFON: '||V_TELEFON);
END LOOP;
CLOSE C_PASAGER;
END;
/
```

-- Sa se afiseze lista cu pasageri a caror stare civila este singur folosing un cursor si o variabila de tip record.

```
DECLARE
CURSOR C_PASAGER IS
SELECT NUME_PASAGER, PRENUME_PASAGER, TELEFON
FROM PASAGERI_BKP WHERE STARE_CIVILA = 'singur';
PASAGER_REC C_PASAGER%ROWTYPE;
BEGIN
OPEN C_PASAGER;
LOOP
FETCH C_PASAGER INTO PASAGER_REC;
EXIT WHEN C_PASAGER%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('NUME: '||PASAGER_REC.NUME_PASAGER||'
PRENUME: '||PASAGER_REC.PRENUME_PASAGER||' TELEFON:
'||PASAGER_REC.TELEFON);
END LOOP;
CLOSE C_PASAGER;
END;
/
```

-- SA SE AFISEZE CELE MAI VIZITATE DESTINATII - DUPA NUMARUL DE BILETE CU ACEA DESTINATIE TOP 3

```
SELECT * FROM DESTINATII;
DECLARE
CURSOR C_DES IS
SELECT D.ORAS,D.DENUMIRE_TARA,COUNT(B.ID_DESTINATIE) NR FROM
BILETE B, DESTINATII D
WHERE D.ID_DESTINATIE = B.ID_DESTINATIE
GROUP BY D.ORAS,D.DENUMIRE_TARA
ORDER BY COUNT(B.ID_DESTINATIE);
```

```
D_REC C_DES%ROWTYPE;
BEGIN
OPEN C_DES;
LOOP
FETCH C_DES INTO D_REC;
EXIT WHEN C_DES%NOTFOUND OR C_DES%ROWCOUNT>3;
DBMS_OUTPUT.PUT_LINE('ORAS: '||D_REC.ORAS||' TARA:
'||D_REC.DENUMIRE_TARA ||' NR: '||D_REC.NR);
END LOOP;
CLOSE C_DES;
END;
/
```

--Folosind un cursor cu parametru, valoarea totala a fiecarei calatorii mai mare decat parametrul dat, cat si orasul.

```
DECLARE
CURSOR c_bilet (p_val BILETE.pret%TYPE) IS
SELECT BI.ID_DESTINATIE, SUM(BI.PRET) VALOARE_CALATORIE,DE.ORAS
FROM BILETE BI, DESTINATII DE
WHERE DE.ID_DESTINATIE=BI.ID_DESTINATIE
GROUP BY BI.ID_DESTINATIE,DE.ORAS
HAVING SUM(BI.PRET)<p_val
ORDER BY VALOARE_CALATORIE DESC;

v_val BILETE.pret%type;
rec_bilet c_bilet%rowtype;
BEGIN

v_val:=1000;
```

```
DBMS_OUTPUT.PUT_LINE('Calatoriile ale caror valoare este mai mica decat parametrul
dat '|| v_val);
IF NOT c_bilet%ISOPEN THEN
OPEN c_bilet (v_val);
END IF;
LOOP
FETCH c_bilet into rec_bilet;
EXIT WHEN c_bilet%notfound;
DBMS_OUTPUT.PUT_LINE('ID_DESTINATIE: '||rec_bilet.ID_DESTINATIE ||'
VALOARE_CALATORIE: '|| rec_bilet.VALOARE_CALATORIE||' ORAS:
'||rec_bilet.ORAS);
END LOOP;
CLOSE c_bilet;
END;
/
```

Exceptii implicite

--Sa se afiseze numele aeroportului din Londra.

```
DECLARE
V_DENUMIRE_AEROPORT DESTINATII.DENUMIRE_AEROPORT%TYPE;
BEGIN
INSERT INTO DESTINATII VALUES (3,'HENRI COANDA','BUCURESTI','REGATUL
UNIT');
SELECT DENUMIRE_AEROPORT INTO V_DENUMIRE_AEROPORT FROM
DESTINATII
WHERE ORAS = 'LONDRA';
DBMS_OUTPUT.PUT_LINE(V_DENUMIRE_AEROPORT);
EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('Exista mai multe aeroporturi');
END;
/
```

--SA SE AFISEZE PRODUCATORUL CU ID-UL 5

```
DECLARE
V_ID PRODUCATORI.ID_PRODUCATOR%TYPE;
BEGIN
SELECT ID_PRODUCATOR INTO V_ID FROM PRODUCATORI;
DBMS_OUTPUT.PUT_LINE(V_ID);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NU S-AU GASIT DATE');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('EXCEPTIE OTHERS');
END;
```

Exceptii explicite

--Sa se modifice tara de origine a producatorului cu id-ul 5

```
DECLARE
ex EXCEPTION;

BEGIN
UPDATE PRODUCATORI
SET TARA_ORIGINE='ROMANIA'
WHERE ID_PRODUCATOR=5;

IF SQL%NOTFOUND THEN
RAISE ex;
END IF;

EXCEPTION
WHEN ex THEN
dbms_output.put_line('Nu exista producatorul cu acest id!');
END;
```

--Daca utilizatorul vrea sa execute blocul PL/SQL in concediu se va declansa o exceptie

```
DECLARE
ex EXCEPTION;

BEGIN
IF SYSDATE > TO_DATE('23-05-2022','DD-MM-YYYY') AND SYSDATE <
TO_DATE('06-06-2022','DD-MM-YYYY') THEN
raise ex;
END IF;

EXCEPTION
WHEN ex THEN
DBMS_OUTPUT.PUT_LINE('Concediu de odihna');
END;
/
```

PACHETE; PROCEDURI; FUNCTII

-- Sa se creeze un pachet cu 2 proceduri si o functie

```
CREATE OR REPLACE PACKAGE PACHET1
IS
PROCEDURE acorda_reducere
(p_clasa IN BILETE.CLASA%TYPE,
procent IN number);
PROCEDURE reda_valoare
(p_clasa IN BILETE.CLASA%TYPE,
p_valoare OUT number);
FUNCTION verifica_valoare
(p_clasa IN BILETE.CLASA%TYPE,
p_clasa2 IN BILETE.CLASA%TYPE) RETURN BOOLEAN;
END PACHET1;
/


CREATE OR REPLACE PACKAGE BODY PACHET1
IS
```

--Sa se acorde o reducere a biletelor cu clasa specificata si un procent dat

```
PROCEDURE acorda_reducere
(p_clasa IN BILETE.CLASA%TYPE,
procent IN number)
IS
BEGIN
UPDATE BILETE
```

```
SET PRET = PRET*(1-procent/100)
WHERE CLASA = P_CLASA;
END acorda_reducere;
```

--Sa se returneze intr-un parametru valoarea totala a biletelor dintr-o categorie specificata

```
PROCEDURE reda_valoare
(p_clasa IN BILETE.CLASA%TYPE,
p_valoare OUT number)
IS
BEGIN
SELECT SUM(BI.PRET) VALOARE_CALATORIE INTO p_valoare FROM BILETE BI
JOIN DESTINATII DE ON DE.ID_DESTINATIE=BI.ID_DESTINATIE
WHERE CLASA = P_CLASA
GROUP BY BI.ID_DESTINATIE,DE.DENUMIRE_AEROPORT;
END reda_valoare;
```

--Sa se returneze true/false daca valoarea totala a biletelor dintr-o clasa este mai mare decat cea a unei alte clase

```
FUNCTION verifica_valoare
(p_clasa IN BILETE.CLASA%TYPE,
p_clasa2 IN BILETE.CLASA%TYPE)
RETURN BOOLEAN
IS
v_val NUMBER;
v_val2 NUMBER;
BEGIN
```

```
reda_valoare(p_clasa,v_val);
reda_valoare(p_clasa2,v_val2);
IF v_val>v_val2 THEN
RETURN TRUE;
ELSE
RETURN FALSE;
END IF;
END;

END PACHET1;
/
```

--Exemple de apel:

```
DECLARE
V_VALOARE NUMBER;
BEGIN
PACHET1.acorda_reducere('I',5);
PACHET1.reda_valoare('Business',V_VALOARE);
DBMS_OUTPUT.PUT_LINE(V_VALOARE);
IF PACHET1.verifica_valoare('I','Business') = TRUE THEN
DBMS_OUTPUT.PUT_LINE('MAI MARE');
ELSE
DBMS_OUTPUT.PUT_LINE('MAI MICA');
END IF;
END;
/
```

-- Sa se creeze al doi-lea pachet cu 2 functii si o procedura

```
CREATE OR REPLACE PACKAGE PACHET2
IS
PROCEDURE AFISARE_LIMITA(LIM_INF BILETE.PRET%TYPE, LIM_SUP
BILETE.PRET%TYPE);
FUNCTION NR_CURSE(AN IN NUMBER) RETURN NUMBER;
FUNCTION tva (valoare IN NUMBER, procent IN NUMBER) RETURN NUMBER;
END PACHET2;
/

CREATE OR REPLACE PACKAGE BODY PACHET2
IS
```

--Sa se afiseze biletele care au costul intre LIM_INF si LIM_SUP de unitati monetare, cat si denumirea pasagerilor si bagajele acestora.

```
PROCEDURE AFISARE_LIMITA(LIM_INF BILETE.PRET%TYPE, LIM_SUP
BILETE.PRET%TYPE) IS
BEGIN
FOR REC_BI IN (SELECT P.ID_PASAGER,
P.NUME_PASAGER,P.TELEFON,P.EMAIL,P.SEX,B.CLASA,B.PRET,BA.STARE_BA
GAJ,BA.CATEGORIE,BA.KILOGRAME FROM BILETE B
LEFT JOIN PASAGERI P ON B.ID_PASAGER=P.ID_PASAGER
LEFT JOIN BAGAJE BA ON B.ID_PASAGER=BA.ID_PASAGER
WHERE PRET BETWEEN LIM_INF AND LIM_SUP)
LOOP
DBMS_OUTPUT.PUT_LINE(REC_BI.NUME_PASAGER||' '||REC_BI.CATEGORIE);
END LOOP;
END AFISARE_LIMITA;
```

-- Sa se afle numarul de curse dintr-un an dat intr-o functie folosind un cursor explicit si comanda %rowcount.

```
FUNCTION NR_CURSE(AN IN NUMBER)
RETURN NUMBER
IS
cursor c is
SELECT BI.DESCRIERE,BI.DATA_PLECARE, BI.CLASA, BI.NUMAR_LOC,
BI.PRET, AV.SERIE, DE.DENUMIRE_AEROPORT FROM BILETE BI
LEFT JOIN DESTINATII DE ON BI.ID_DESTINATIE=DE.ID_DESTINATIE
LEFT JOIN AVIOANE AV ON BI.ID_AVION=AV.ID_AVION
WHERE EXTRACT(YEAR FROM BI.DATA_PLECARE) = AN;
cnt number;
BEGIN
OPEN c;
cnt:=c%ROWCOUNT;
CLOSE c;
return cnt;
END;
```

-- Sa se realizeze o functie care calculeaza tva-ul unui bilet

```
FUNCTION tva (valoare IN NUMBER, procent IN NUMBER)
RETURN NUMBER IS
BEGIN
RETURN (valoare*procent);
END tva;


END PACHET2;
/
```

Triggeri

-- sa se creeze un trigger care se declanseaza dupa fiecare inregistrare in tabela avioane

```
create or replace trigger t_adauga_avioane
after insert on avioane
begin
DBMS_OUTPUT.PUT_LINE('Triggerul s-a executat. S-a adaugat o inregistrare noua! ');
end;
/
```

-- sa se creeze un trigger care se declanseaza la stergerea unei inregistrari din tabela destinatii si va returna o eroare

```
create or replace trigger t_sterge_destinatii
before delete on destinatii
begin
RAISE_APPLICATION_ERROR(-20000, 'Nu poti sterge inregistrarile.');
end;
/
```

-- sa se creeze un trigger care se declanseaza in momentul in care vrei sa faci update la pretul unui bilet

-- cu o suma mai mica de 300, atunci cand vechiul pret e mai mare de 500

```
create or replace trigger update_bilete
before update on bilete
for each row
when(old.pret>500)
begin
if(:new.pret <300)then
RAISE_APPLICATION_ERROR(-28000, 'Pentu biletele cu pretul mai mare de 500, nu se poate modifica pretul cu o valoare mai mica de 300.');
end if;
end;
/
```