# **Proiect baze de date**

# **P-10**

# **Vasile Gila-Roxana**

# **Grupa 6**

Proiect la Baze de date - 10

Se considera o aplicație pentru evidența cheltuielilor unei firme, care efectuează plăți în valută

(registru de casa). Baza de date Oracle conține următoarele informații:

- curs valutar: data, rata de schimb

- parteneri: nume, CUI, Adresa

- tranzacții: data efectuării, CUI, suma în valuta, suma în lei

- bilanț zilnic: data, sold inițial, total intrări, total ieșiri, sold final.

Știind că rata de schimb este cuprinsa între 1.00 si 4.00, numele nu depășește 10 caractere,

CUI are exact 6 caractere, suma este negativă dacă se plătește partenerului și pozitivă dacă se

încasează de la partener, și că soldul inițial a fost 0, se cere:

1. Să se realizeze proiectarea bazei de date aferente (structura de tabele, structura de

coloane a fiecărei tabele, constrângeri).

2. Să se scrie comenzile SQL pentru tabelele proiectate la punctul anterior.

3. Să se scrie comenzile SQL pentru popularea bazei de date cu 5 tranzacții efectuate

considerând 3 parteneri, in 2 zile consecutive.

4. Să se implementeze un mecanism care să nu permită modificarea cursului valutar

pentru zilele când sunt efectuate tranzacții, să se argumenteze alegerea făcută.

5. Să se scrie o procedura care să genereze un raport care să cuprindă numele, CUI,

intrări, ieșiri pentru partenerul specificat ca și argument prin CUI.

6. Să se genereze un raport detaliat care să cuprindă pentru fiecare partener: numele,

CUI-ul, și toate tranzacțiile efectuate (data tranzacție, tip tranzacție (intrare sau ieșire),

suma în valuta, suma în lei), ordonat după nume și data efectuării tranzacțiilor.

7. Să se scrie un trigger care la adăugarea unei tranzacții să calculeze automat suma în

lei, iar daca cursul nu este specificat, să semnaleze eroare și să nu permită adăugarea

tranzacției.

8. Să se scrie o funcție care să primească ca și parametru CUI și să returneze rulajul

efectuat de acel partener (suma dintre intrări și ieșiri luate fără semn).

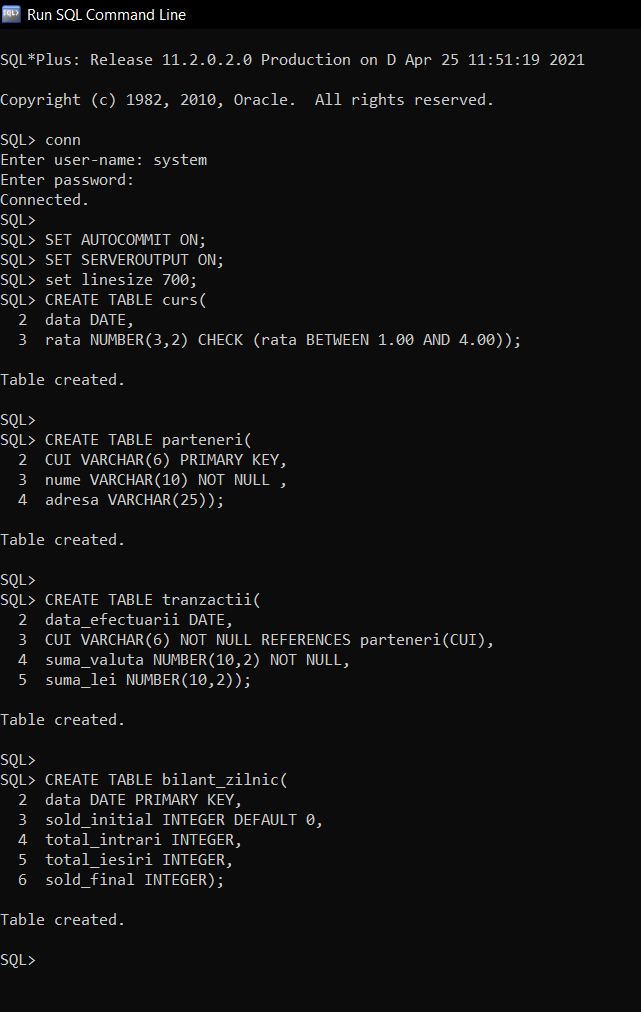
9. Să se afișeze partenerii care doar au încasat și care nu au făcut plăți într-un an întreg,

precizând: nume, CUI, suma totala încasată.

10. Să se afișeze partenerul care are cele mai multe tranzacții, precizând numele, CUI-ul,

număr de tranzacții, rata lui de rentabilitate (cât la sută este profit din suma totală a

tranzacțiilor).



**1. Să se realizeze proiectarea bazei de date aferente (structura de tabele, structura de**

**coloane a fiecărei tabele, constrângeri).**

Pentru a-mi fi mai usor să rezolv problema am împărțit informațiile ce trebuie stocate în baza de date în patru tabele: curs, parteneri, tranzactii, bilant\_zilnic; fiecare conținând elementele specifice ce reies din enunțul problemei. Tabelele tranzactii și parteneri sunt interconectate, legătura fiind făcută prin CUI.

**2. Să se scrie comenzile SQL pentru tabelele proiectate la punctul anterior.**

SET AUTOCOMMIT ON;

SET SERVEROUTPUT ON;

CREATE TABLE curs(

data DATE,

rata NUMBER(3,2) CHECK (rata BETWEEN 1.00 AND 4.00));

CREATE TABLE parteneri(

CUI VARCHAR(6) PRIMARY KEY,

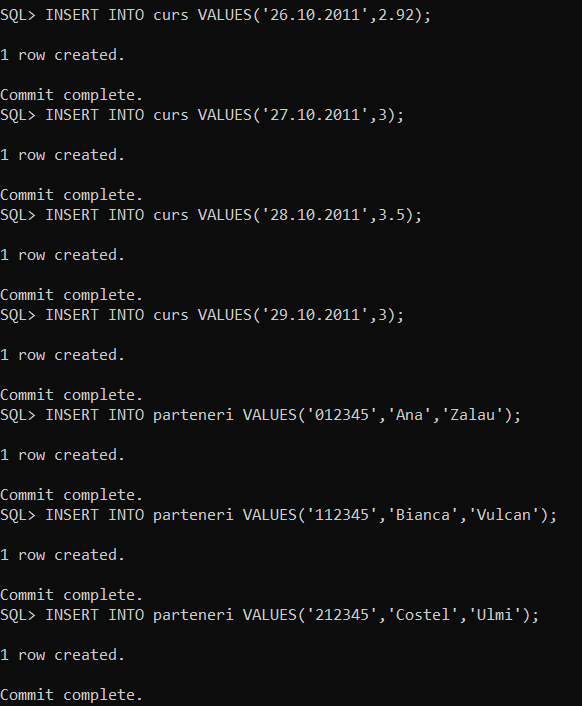
nume VARCHAR(10) NOT NULL ,

adresa VARCHAR(25));

CREATE TABLE tranzactii(

data\_efectuarii DATE,

CUI VARCHAR(6) NOT NULL REFERENCES parteneri(CUI),

suma\_valuta NUMBER(10,2) NOT NULL,

suma\_lei NUMBER(10,2));

CREATE TABLE bilant\_zilnic(

data DATE PRIMARY KEY,

sold\_initial INTEGER DEFAULT 0,

total\_intrari INTEGER,

total\_iesiri INTEGER,

sold\_final INTEGER);

**3. Să se scrie comenzile SQL pentru popularea bazei de date cu 5 tranzacții efectuate**

**considerând 3 parteneri, in 2 zile consecutive.**

INSERT INTO curs VALUES('26.10.2011',2.92);

INSERT INTO curs VALUES('27.10.2011',3);

INSERT INTO curs VALUES('28.10.2011',3.5);

INSERT INTO curs VALUES('29.10.2011',3);

INSERT INTO parteneri VALUES('012345','Ana','Zalau');

INSERT INTO parteneri VALUES('112345','Bianca','Vulcan');

INSERT INTO parteneri VALUES('212345','Costel','Ulmi');

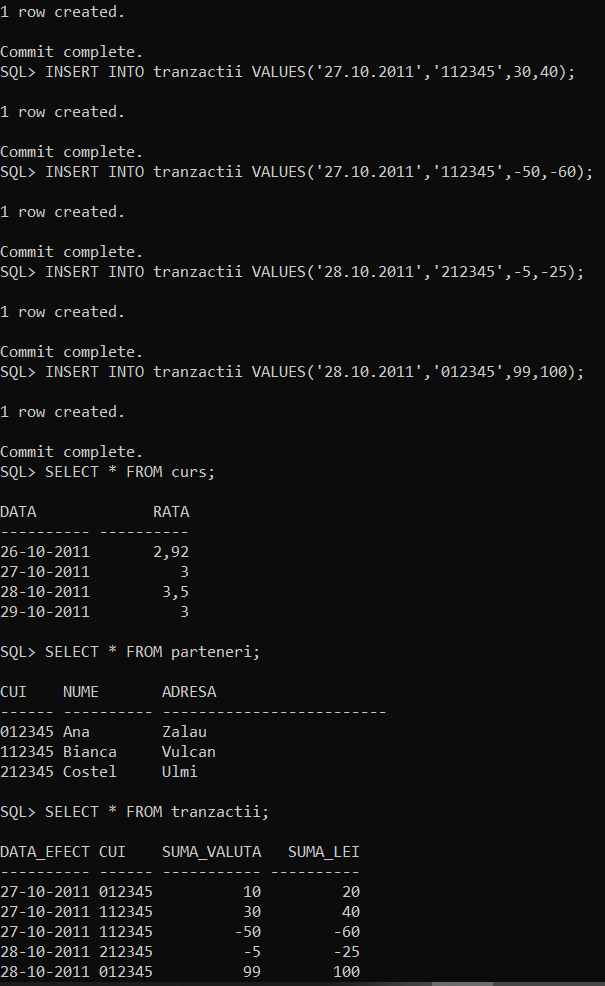
INSERT INTO tranzactii VALUES('27.10.2011','012345',10,20);

INSERT INTO tranzactii VALUES('27.10.2011','112345',30,40);

INSERT INTO tranzactii VALUES('27.10.2011','112345',-50,-60);

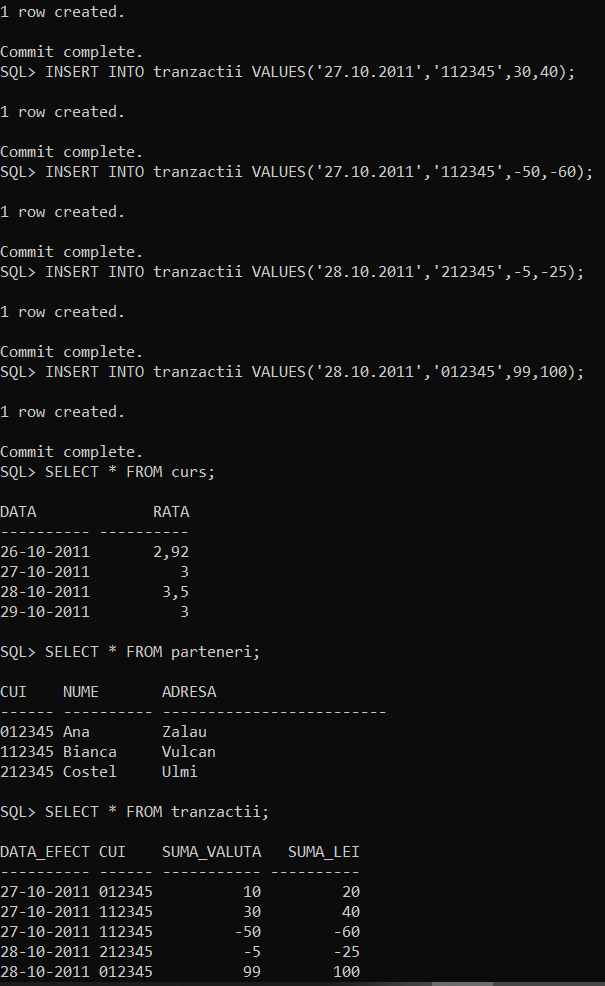
INSERT INTO tranzactii VALUES('28.10.2011','212345',-5,-25);

INSERT INTO tranzactii VALUES('28.10.2011','012345',99,100);

SELECT \* FROM curs;

SELECT \* FROM parteneri;

SELECT \* FROM tranzactii;



**4. Să se implementeze un mecanism care să nu permită modificarea cursului valutar**

**pentru zilele când sunt efectuate tranzacții, să se argumenteze alegerea făcută.**

Pentru rezolvarea cerinței am decis să creez un trigger. Am ales această metodă de rezolvare deoarece mi s-a părut cea mai eficientă pentru rezolvarea cerinței și totodată este ușor de implementat. Trigger-ul are următorul algoritm: înainte de orice modificare asupra tabelei curs (acest fapt aplicându-se numai asupra modificărilor ce doresc a fi efectuate după crearea trigger-ului) se alocă variabilei noi declarate nrdata de cate ori data asupra căreia se dorește modificarea există în tabela tranzacții; daca nrdata este diferit de zero, și deci există tranzacții efectuate în data respectivă, triggerul semnalează eroare și modificarea nu se efectuează.

CREATE OR REPLACE TRIGGER modif\_curs4

BEFORE UPDATE ON curs

FOR EACH ROW

DECLARE

nrdata INTEGER;

BEGIN

SELECT COUNT(data\_efectuarii) INTO nrdata FROM tranzactii WHERE data\_efectuarii=:new.data;

IF nrdata<>0 THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu se poate modifica rata! S-au efectuat tranzactii in data introdusa!');

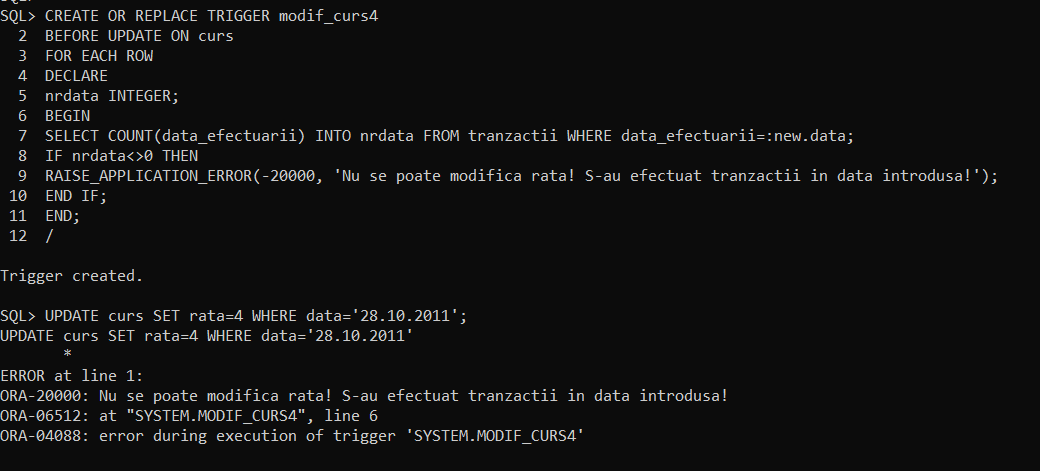
END IF;

END;

/

UPDATE curs SET rata=4 WHERE data='28.10.2011';

Pentru dovedirea funcționării am încercat modificarea asupra ratei de schimb pentru o dată în care s-au efectuat tranzacții(aceasta nu s-a efectuat, s-a afișat mesajul de eroare).



**5. Să se scrie o procedura care să genereze un raport care să cuprindă numele, CUI,**

**intrări, ieșiri pentru partenerul specificat ca și argument prin CUI.**

Procedura are următorii pași: selecteaza numele clientului ce are CUI-ul dat ca parametru și îl inserează în variabila nume1 declarată în cadrul său, în variabila intrari/iesiri este inserată suma intrărilor/ieșirilor efectuate de client ce se gasesc în tabela tranzacții; Se verifică pe rând dacă variabilele intrari, iesiri sunt nule, iar în caz afirmativ li se atribuie valoarea 0 (am făcut asta pentru ca la afisare să nu rămană gol unde trebuia valorea variabilei); Se afișează informațiile cerute cu ajutorul funcției PUT\_LINE a pachetului predefinit DBMS\_OUTPUT.

CREATE OR REPLACE PROCEDURE raport1(cui1 varchar)

AS

nume1 VARCHAR(10);

intrari INTEGER;

iesiri INTEGER;

BEGIN

SELECT nume INTO nume1 FROM parteneri WHERE cui=cui1;

SELECT SUM(suma\_valuta) INTO intrari FROM tranzactii WHERE cui=cui1 AND suma\_valuta>0;

SELECT ABS(SUM(suma\_valuta)) INTO iesiri FROM tranzactii WHERE cui=cui1 AND suma\_valuta<0;

IF intrari IS NULL THEN

intrari:=0;

END IF;

IF iesiri IS NULL THEN

iesiri:=0;

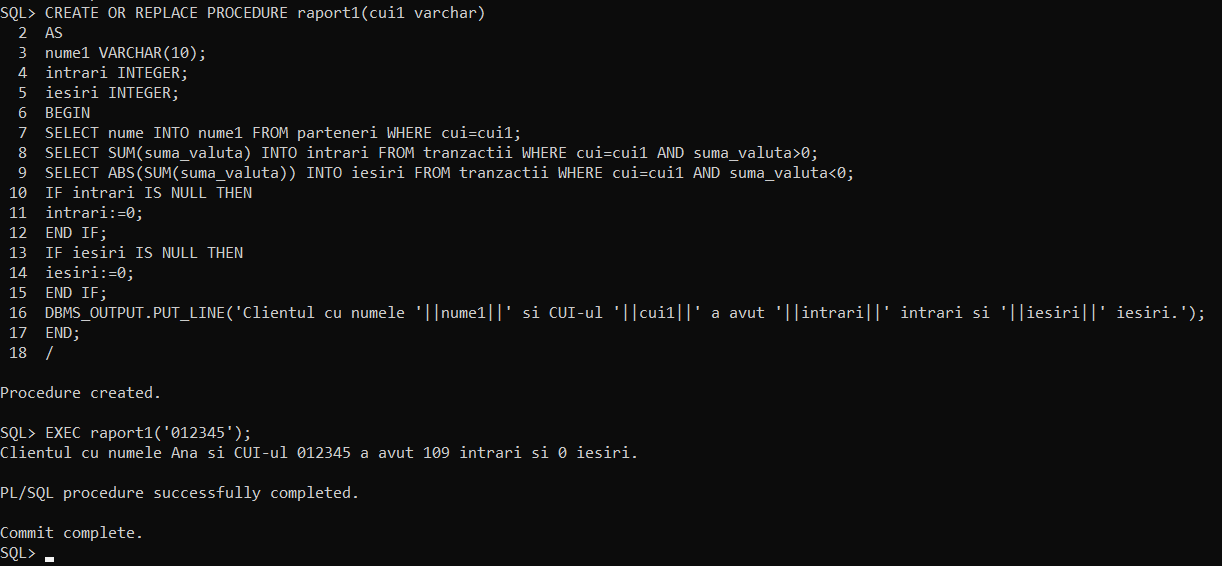
END IF;

DBMS\_OUTPUT.PUT\_LINE('Clientul cu numele '||nume1||' si CUI-ul '||cui1||' a avut '||intrari||' intrari si '||iesiri||' iesiri.');

END;

/

EXEC raport1('012345');



**6. Să se genereze un raport detaliat care să cuprindă pentru fiecare partener: numele,**

**CUI-ul, și toate tranzacțiile efectuate (data tranzacție, tip tranzacție (intrare sau ieșire),**

**suma în valuta, suma în lei), ordonat după nume și data efectuării tranzacțiilor.**

Pentru a simplifica metoda de identificare a tipului de tranzacție am folosit operatiția UNION între doua SELECT-uri, unul fiind pentru selectarea intrărilor, iar celălalt pentru ieșiri.

SELECT nume, A.CUI, data\_efectuarii, suma\_valuta, suma\_lei, 'intrare' AS tip\_tranzactie

FROM tranzactii A, parteneri B

WHERE suma\_valuta>0 AND A.CUI=B.CUI

GROUP BY nume, A.CUI, data\_efectuarii, suma\_valuta, suma\_lei

UNION

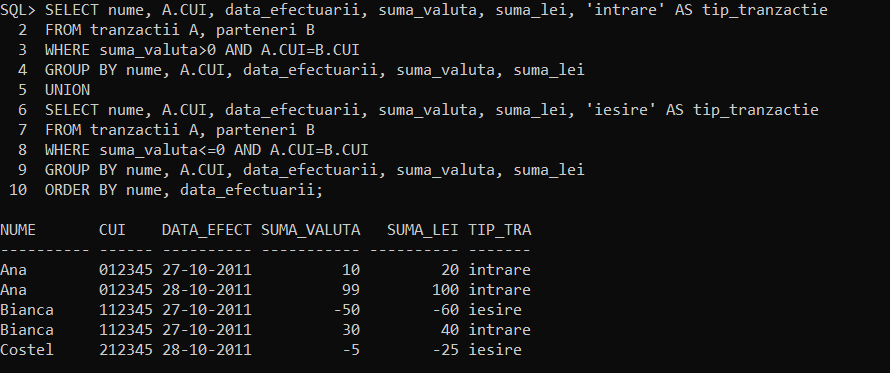
SELECT nume, A.CUI, data\_efectuarii, suma\_valuta, suma\_lei, 'iesire' AS tip\_tranzactie

FROM tranzactii A, parteneri B

WHERE suma\_valuta<=0 AND A.CUI=B.CUI

GROUP BY nume, A.CUI, data\_efectuarii, suma\_valuta, suma\_lei

ORDER BY nume, data\_efectuarii;



**7. Să se scrie un trigger care la adăugarea unei tranzacții să calculeze automat suma în**

**lei, iar daca cursul nu este specificat, să semnaleze eroare și să nu permită adăugarea**

**tranzacției.**

Algoritmul trigger-ului este: inserează în variabila nrdata de câte ori data respectivă se regăsește în tablela curs (maxim 1); dacă nrdata este diferită de 0, înseamnă că este specificată rata de schimb pentru data respectivă și i se atribuie variabilei suma\_lei valorea ratei de schimb înmulțită cu suma in valută; în cazul în care nrdata este 0 (deci nu este specificat cursul în ziua respectivă) se semnalează eroare și adaugarea nu are loc.

CREATE OR REPLACE TRIGGER suma

BEFORE INSERT ON tranzactii

FOR EACH ROW

DECLARE

nrdata INTEGER;

r\_curs NUMBER(3,2);

BEGIN

SELECT COUNT(data) INTO nrdata FROM curs WHERE data=:NEW.data\_efectuarii;

IF nrdata<>0 THEN

SELECT rata INTO r\_curs FROM curs WHERE data=:NEW.data\_efectuarii;

:NEW.suma\_lei:=:NEW.suma\_valuta\*r\_curs;

ELSE

RAISE\_APPLICATION\_ERROR(-20000, 'Rata de schimb inexistenta pentru ziua specificata!');

END IF;

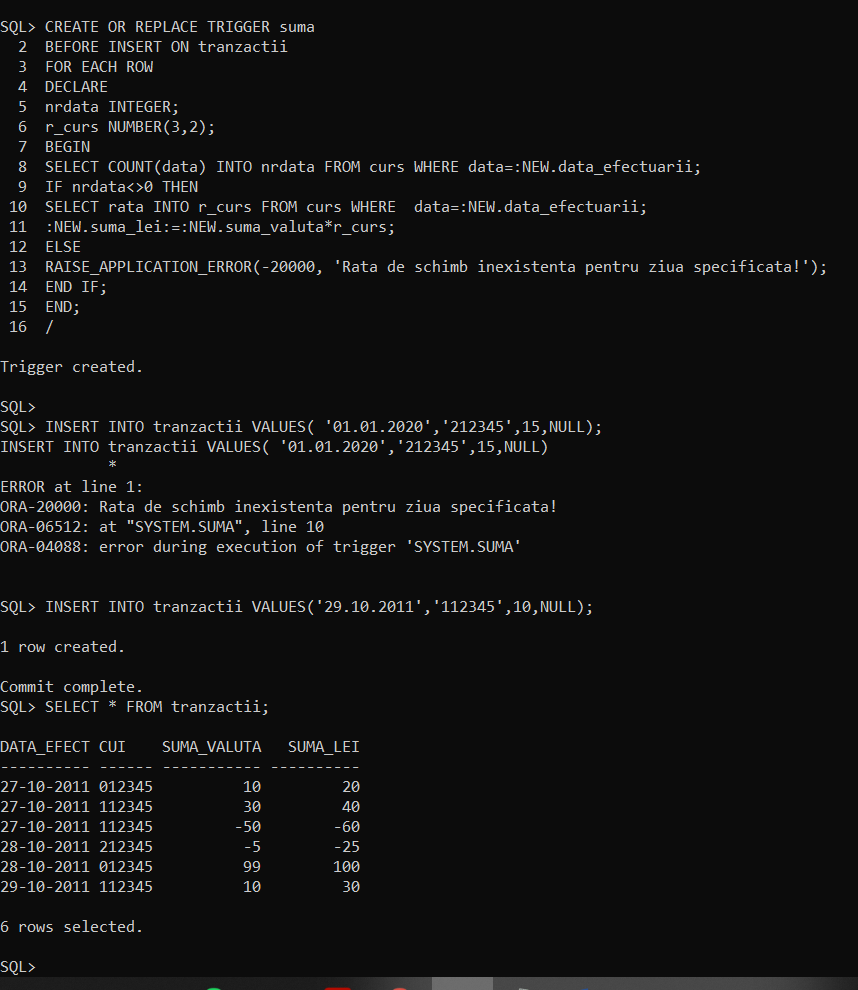
END;

/

INSERT INTO tranzactii VALUES( '01.01.2020','212345',15,NULL);

INSERT INTO tranzactii VALUES('29.10.2011','112345',10,NULL);

SELECT \* FROM tranzactii;



**8. Să se scrie o funcție care să primească ca și parametru CUI și să returneze rulajul**

**efectuat de acel partener (suma dintre intrări și ieșiri luate fără semn).**

Cu ajutorul funcțiilor SUM(pentru calcularea sumei intrărilor/ieșirilor) și ABS(pentru modului ieșirilor) funcția salvează în variabilele intrări și ieșiri cele doua elemente care însumate reprezintă rulajul efectuat de partenerul specificat prin CUI (ce este dat ca și parametru). În cazul în care partenerul nu a avut intrări/ieșiri, variabilei corespunzătoare i se atribuie valoarea 0 pentru a elimina eventualele erori ale rezultatului (de exemplu dacă cel puțin una dintre valorile adunate este egală cu null, rezultatul este de asemenea null).

CREATE OR REPLACE FUNCTION rulaj(cui1 varchar)

RETURN INTEGER AS

intrari INTEGER;

iesiri INTEGER;

BEGIN

SELECT SUM(suma\_valuta) INTO intrari FROM tranzactii WHERE cui=cui1 AND suma\_valuta>0;

SELECT ABS(SUM(suma\_valuta)) INTO iesiri FROM tranzactii WHERE cui=cui1 AND suma\_valuta<0;

IF intrari IS NULL THEN

intrari:=0;

END IF;

IF iesiri IS NULL THEN

iesiri:=0;

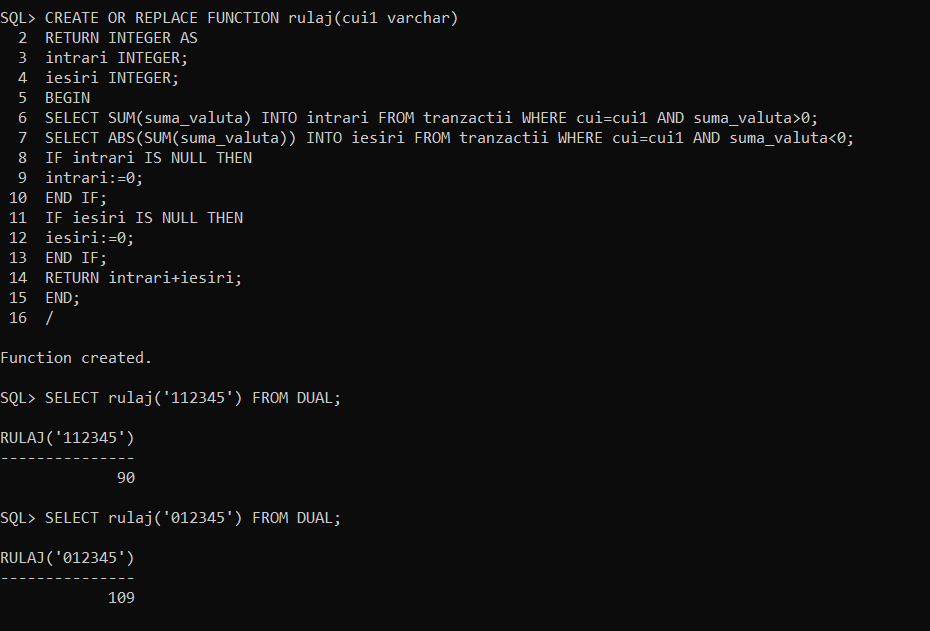
END IF;

RETURN intrari+iesiri;

END;

/

SELECT rulaj('012345') FROM DUAL;



**9. Să se afișeze partenerii care doar au încasat și care nu au făcut plăți într-un an întreg,**

**precizând: nume, CUI, suma totala încasată.**

Pentru o mai bună exemplificare a rezultatului comenzii SELECT am introdus câte o valore în tabela curs, respectiv tranzactii. Această interogare pe două tabele folosește funcția de la punctul 8 ( funcția rulaj ) pentru a verifica dacă rulajul efectuat de un respectiv partener este egal cu suma încasărilor, deci prin urmare acesta doar a încasat și nu a făcut nici o plata. De asemenea, se utilizează funcția MONTHS\_BETWEEN cu ajutorul căreia se calculează lunile dintre data actuală și data ultimei tranzacții efectuate de un anume partener, astfel valorificându-se cerința ca să nu se fi făcut plăți într-un an întreg.

INSERT INTO curs VALUES('29.03.2020',3);

INSERT INTO tranzactii VALUES('29.03.2020','012345',10,NULL);

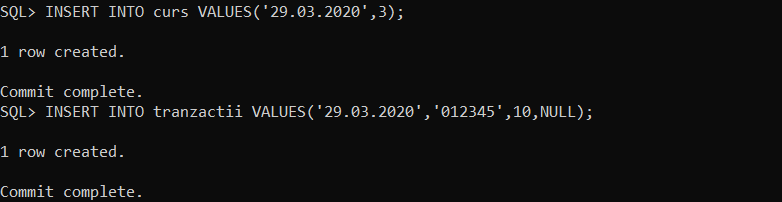
SELECT nume, A.CUI, SUM(suma\_valuta) AS suma\_totala

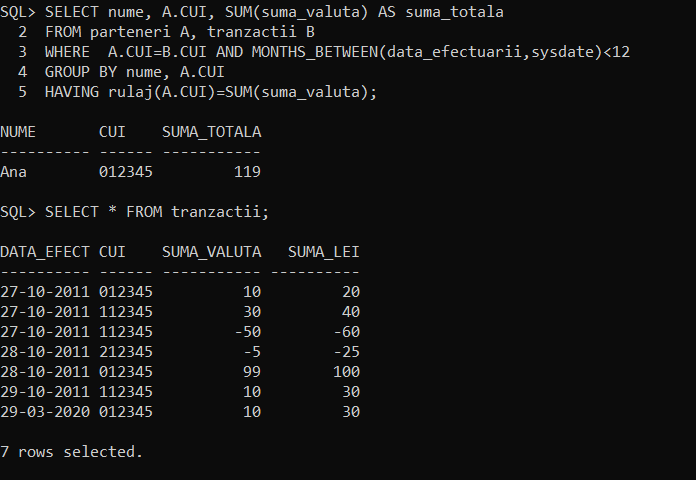
FROM parteneri A, tranzactii B

WHERE A.CUI=B.CUI AND MONTHS\_BETWEEN(data\_efectuarii,sysdate)<=12

GROUP BY nume, A.CUI

HAVING rulaj(A.CUI)=SUM(suma\_valuta);





PS. Am realizat târziu că trebuia <=12 (nu <12) și de aceea nu apare așa în captura de ecran.

**10. Să se afișeze partenerul care are cele mai multe tranzacții, precizând numele, CUI-ul,**

**număr de tranzacții, rata lui de rentabilitate (cât la sută este profit din suma totală a**

**tranzacțiilor).**

Interogarea folosește funcția COUNT pentru calcula numărul de tranzacții, folosind un alias pentru acesta (nr\_tranzactii). Se utlizează și funcția NVL care returnează suma intrărilor dacă acestea există, respectiv 0 în caz contrar. De asmenea, pentru a ușura calculul ratei de rentabilitate se utlizează funcția rulaj de la punctul 8. În clauza HAVING este verificată condiția ca partenerul cu CUI-ul respectiv să fie cel cu cele mai multe tranzacții (utilizându-se selecarea nr-ului maxim de tranzacții efectuate de un partener cu ajutorului funcției MAX).

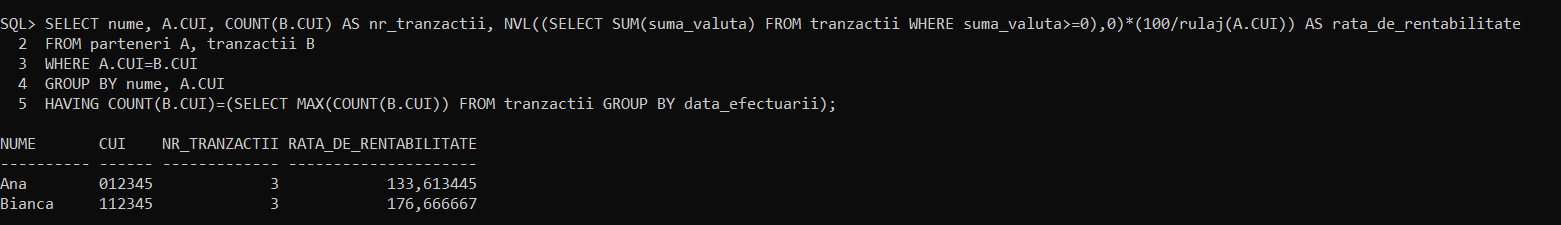
SELECT nume, A.CUI, COUNT(B.CUI) AS nr\_tranzactii, NVL((SELECT SUM(suma\_valuta) FROM tranzactii WHERE suma\_valuta>=0),0)\*(100/rulaj(A.CUI)) AS rata\_de\_rentabilitate

FROM parteneri A, tranzactii B

WHERE A.CUI=B.CUI

GROUP BY nume, A.CUI

HAVING COUNT(B.CUI)=(SELECT MAX(COUNT(B.CUI)) FROM tranzactii GROUP BY data\_efectuarii);



**PS.**

După ce am terminat rezolvarea problemei, am realizat că am ”ignorat” tabela bilant\_zilnic. Prin urmare, am făcut un trigger care face ca atunci când se înserează sau se modifică tabela tranzactii,să se facă de asemenea inserări sau modificări asupra tabelei bilant\_zilnic. Pentru exemplificare, am făcut 3 inserări în tabela tranzacții.

CREATE OR REPLACE TRIGGER bilant

AFTER INSERT OR UPDATE ON tranzactii

FOR EACH ROW

DECLARE

data1 INTEGER;

BEGIN

SELECT COUNT(data) INTO data1 FROM bilant\_zilnic WHERE data=:NEW.data\_efectuarii;

IF data1<>0 THEN

IF :NEW.suma\_valuta<0 THEN

UPDATE bilant\_zilnic SET total\_iesiri=total\_iesiri+:NEW.suma\_valuta WHERE data=:NEW.data\_efectuarii;

ELSE

UPDATE bilant\_zilnic SET total\_intrari=total\_intrari+:NEW.suma\_valuta WHERE data=:NEW.data\_efectuarii;

END IF;

UPDATE bilant\_zilnic SET sold\_final=sold\_final+:NEW.suma\_valuta WHERE data=:NEW.data\_efectuarii;

ELSE

IF :NEW.suma\_valuta<0 THEN

INSERT INTO bilant\_zilnic(data,sold\_initial,total\_intrari,total\_iesiri,sold\_final) VALUES(:NEW.data\_efectuarii, 0, 0, :NEW.suma\_valuta, :NEW.suma\_valuta);

ELSE

INSERT INTO bilant\_zilnic(data,sold\_initial,total\_intrari,total\_iesiri,sold\_final) VALUES(:NEW.data\_efectuarii,0 , :NEW.suma\_valuta,0,:NEW.suma\_valuta);

END IF;

END IF;

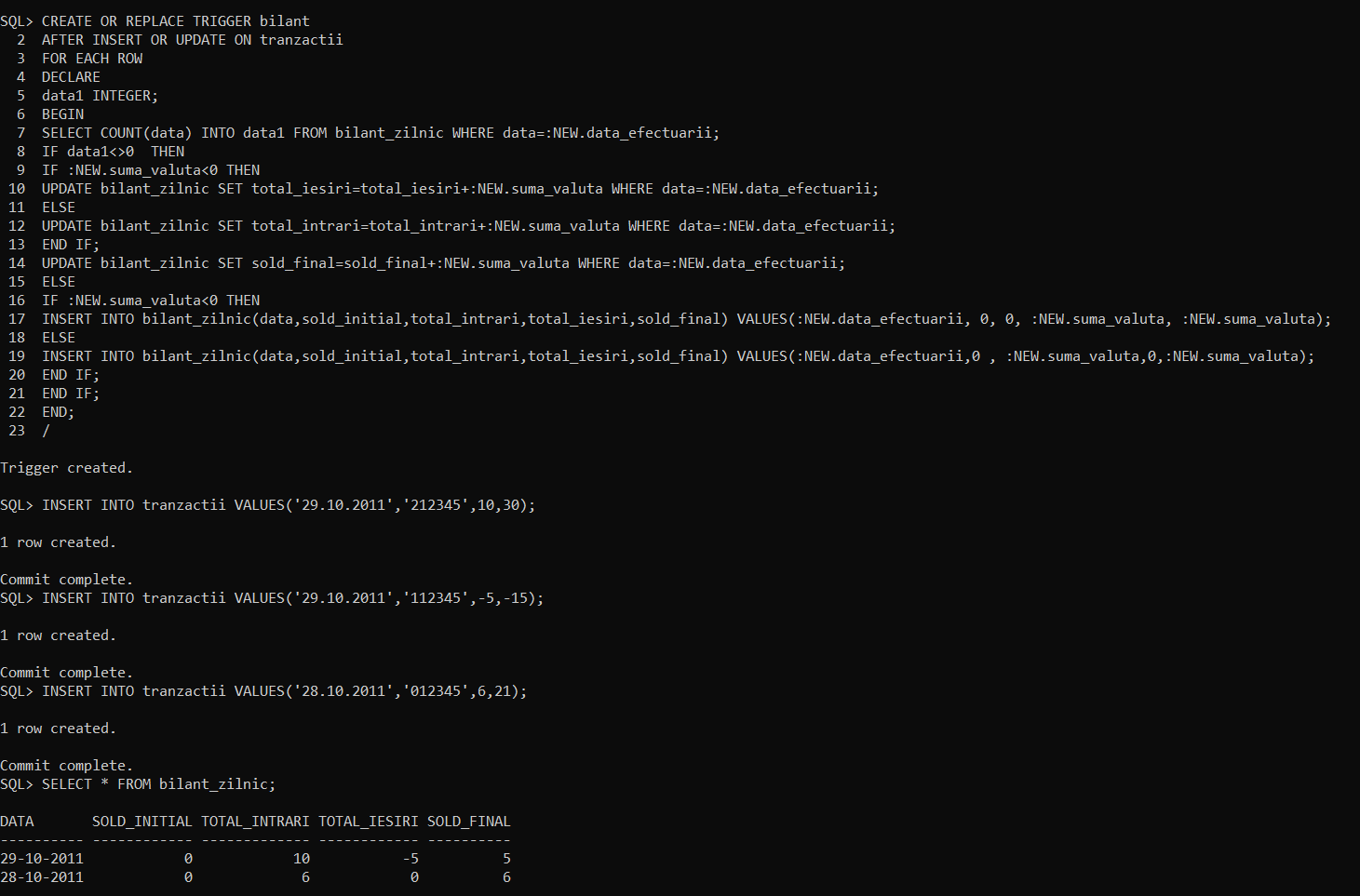
END;

/

INSERT INTO tranzactii VALUES('29.10.2011','212345',10,30);

INSERT INTO tranzactii VALUES('29.10.2011','112345',-5,-15);

INSERT INTO tranzactii VALUES('28.10.2011','012345',6,21);



Trigger-ul funcționează astfel:

* Folosește funcția COUNT pentru a număra de câte ori apare noua data, ce este inserata în tabela tranzactii, și tabela bilant\_zilnic;
* Dacă numărul de apariții al datei noi este diferit de 0, ceea ce înseamnă că au mai fost făcute tranzacții în ziua respectivă, atunci trigger-ul actualizează valorile pentru total\_iesiri, total\_intrari și sold\_final, în consecință pentru fiecare;
* Altfel, daca în ziua respectivă nu s-au făcut tranzacții până în acest moment (data1=0), trigger-ul înserează în tabela bilant\_zilnic noile valori în conformitate cu datele inserate în tabela tranzactii;
* În fiecare din situațiile în care se poate afla numărul de apariții al datei noi inserate, se verifică dacă valorea variabilei suma\_valuta este negativă sau pozitivă pentru a se putea efectua inserări/modificări în consecință.