Universitatea Politehnica Timișoara

Facultatea de Automatică și Calculatoare

Secția Ingineria Sistemelor

Proiect la Baze de date

- TEMA 2 -

GODEANU LUCIAN-FLORIN

AC-IS, AN II

GRUPA 3.2

Enuntul proiectului

Se considera o aplicatie online care este utilizata pentru realizarea bilantului contabil al unei firme. Aplicatia foloseste pentru stocarea datelor o baza de date Oracle (considerata ca o colectie de tabele) care trebuie sa contina informatiile enumerate mai jos:

- numar cont – maxim 5 cifre,
- descriere cont – maxim 50 de caractere,
- tip cont – 2 caractere (’ca’ –capital, ’pa’-pasiv, ’ac’-activ),
- sold_initial (suma disponibila in cont la inceputul anului curent),
- sold (suma disponibila in cont la momentul curent),
- numar tranzactie (nu poate fi mai mare decat 10000),
- data tranzactiei – data calendaristica,
- suma tranzactionata (pentru un cont, intr-o singura tranzactie, nu poate fi mai mare decat 10000),
- numar cont creditor – (contul din care pleaca banii) maxim 5 cifre
- numar cont debitor – (contul in care intra banii) maxim 5 cifre
- descriere tranzactie – maxim 10 caractere.

Cerintele proiectului

Cerinta 1:

Sa se proiecteze baza de date, scriindu-se comenzile de creare a tabelelor si impunandu-se toate constrangerile care sunt considerate utile din punct de vedere functional si al integritatii datelor.

Cod SQL:

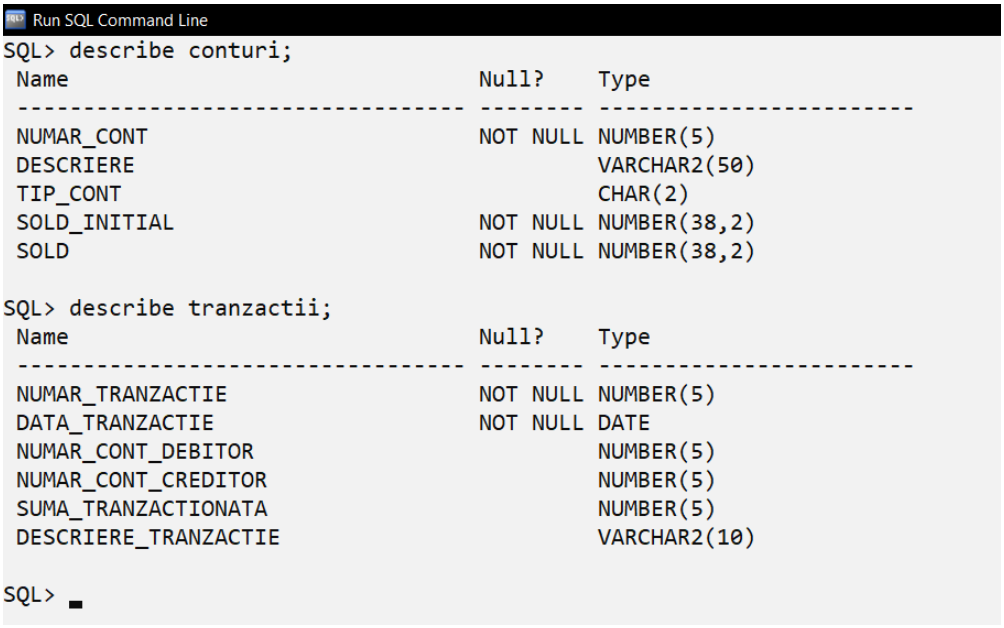
```
SET AUTOCOMMIT ON;
set linesize 300;
select sysdate from dual;

CREATE TABLE conturi (
numar_cont NUMBER (5) primary key,
descriere VARCHAR (50),
tip_cont CHAR (2) check ((tip_cont = 'ca') OR (tip_cont = 'pa') OR
(tip_cont = 'ac')),
sold_initial NUMBER (*, 2) not null,
sold NUMBER (*, 2) not null
);

CREATE TABLE tranzactii (
numar_tranzactie NUMBER (5) primary key check (numar_tranzactie <= 10000),
data_tranzactie DATE not null,
numar_cont_debitor NUMBER (5) referencing conturi (numar_cont) on delete
cascade,
numar_cont_creditor NUMBER (5) referencing conturi (numar_cont) on delete
cascade,
suma_tranzactionata NUMBER (5) check (suma_tranzactionata <= 10000),
descriere_tranzactie VARCHAR (10)
);

describe    conturi;
describe tranzactii;
```

Implementare:



```
Run SQL Command Line
SQL> describe conturi;
 Name                              Null?    Type
 --------------------------------- -------- ------------------------
 NUMAR_CONT                        NOT NULL NUMBER(5)
 DESCRIERE                                  VARCHAR2(50)
 TIP_CONT                                   CHAR(2)
 SOLD_INITIAL                      NOT NULL NUMBER(38,2)
 SOLD                              NOT NULL NUMBER(38,2)

SQL> describe tranzactii;
 Name                              Null?    Type
 --------------------------------- -------- ------------------------
 NUMAR_TRANZACTIE                  NOT NULL NUMBER(5)
 DATA_TRANZACTIE                   NOT NULL DATE
 NUMAR_CONT_DEBITOR                         NUMBER(5)
 NUMAR_CONT_CREDITOR                        NUMBER(5)
 SUMA_TRANZACTIONATA                        NUMBER(5)
 DESCRIERE_TRANZACTIE                       VARCHAR2(10)

SQL>
```

Explicatii:

Am creat 2 tabele pentru a realiza bilantul contabil al unei firme. Astfel, am construit o tabela ‘conturi’ unde vor fi stocate datele initiale ale conturilor clientilor, dar si o tabela ‘tranzactii’ pentru a putea realiza actiuni intre contul creditor (cont din care pleaca banii) si contul debitor (contul in care intra banii).

Am folosit constrangeri precum:

Constrangerea de tip NOT NULL ce asigura faptul ca o coloana sa nu contina valoarea nula. Ea poate fi specificata la nivel de coloana si nu la nivel de tabel.

Constrangerea PRIMARY KEY ce creeaza o cheie primara pentru tabel-numar_cont. O singura cheie poate fi creata pentru fiecare tabel. Aceasta constrangere este o coloana sau un set de coloane care identifica in mod unic fiecare rand al tabelului. Nici o coloana care face parte din cheia primara nu poate contine valoarea NULL. Poate fi definita la nivel de coloana sau tabel.

Constrangerea de tip CHECK defineste o conditie pe care fiecare rand trebuie sa o satisfaca- numar_tranzactie <= 10000 sau tipurile de cont (ca/pa/ac).

Clauza ON DELETE CASCADE indica faptul ca, atunci cand o linie din tabela ‘parinte’ (tabela conturi) este stearsa, vor fi sterse si liniile dependente din tabela ‘copil’ (tabela tranzactii).

Cerinta 2:

Sa se scrie comenzile de populare a tabelelor cu informatii (minim 5 articole in fiecare tabela).

Cod SQL:

```
INSERT INTO conturi VALUES (42266, 'Irina Deaconescu', 'ac', 14226,
19942);
INSERT INTO conturi VALUES (22520, 'Godeanu Lucian', 'ca', 22520, 0);
INSERT INTO conturi VALUES (69420, 'Mirela Ionela', 'ac', 6942, 10565);
INSERT INTO conturi VALUES (18069, 'Brailoiu Razvan', 'pa', 34226, 0);
INSERT INTO conturi VALUES (33370, 'Cristina Maria', 'ca', 3700, 3700);

INSERT INTO tranzactii VALUES (111, '04-06-2018', 42266, 22520, 8805,
'InelAur');
INSERT INTO tranzactii VALUES (222, '22-05-2019', 22520, 69420, 9688,
'Porsche911');
INSERT INTO tranzactii VALUES (333, '16-12-2020', 69420, 18069, 3336,
'PlataClub');
INSERT INTO tranzactii VALUES (444, '18-09-2021', 18069, 33370, 8187,
'Investitii');
INSERT INTO tranzactii VALUES (555, '03-03-2022', 33370, 42266, 1989,
'Chirie');

select*from conturi;
select*from tranzactii;
```

Implementare:

Select Run SQL Command Line

```
SQL> set linesize 1000;
SQL> select*from conturi;

NUMAR_CONT DESCRIERE                                          TI SOLD_INITIAL       SOLD
---------- -------------------------------------------------- -- ------------ ----------
     42266 Irina Deaconescu                                   ac        14226      19942
     22520 Godeanu Lucian                                     ca        22520          0
     69420 Mirela Ionela                                      ac         6942      10565
     18069 Brailoiu Razvan                                    pa        34226          0
     33370 Cristina Maria                                     ca         3700       3700

SQL> select*from tranzactii;

NUMAR_TRANZACTIE DATA_TRANZ NUMAR_CONT_DEBITOR NUMAR_CONT_CREDITOR SUMA_TRANZACTIONATA DESCRIERE_
---------------- ---------- ------------------ -------------------- ------------------- ----------
             111 04-06-2018              42266                22520                8805 InelAur
             222 22-05-2019              22520                69420                9688 Porsche911
             333 16-12-2020              69420                18069                3336 PlataClub
             444 18-09-2021              18069                33370                8187 Investitii
             555 03-03-2022              33370                42266                1989 Chirie

SQL>
```

Explicatii:

Pentru fiecare tabela, am introdus cate 6 articole, folosind instructiunea SET LINESIZE 1000 pentru a imi afisa toate campurile in linie. Cu ajutorul functiei ‘INSERT INTO … VALUES (…)’ am introdus articolele, completand cu datele aferente si potrivite comenzilor de creare a tabelelor.

- ***Cerinta 3:***

Sa se implementeze o procedura stocata care sa implementeze o tranzactie, stiind ca aceasta presupune 2 conturi – unul debitor (din care pleaca banii) si respectiv unul creditor (in care intra banii). Procedura primeste ca si parametri de intrare numarul tranzactiei, contul debitor, contul credtor, suma tranzactionata precum si o descriere a tranzactiei.

Cod SQL:

```
CREATE OR REPLACE PROCEDURE tranzactie (parametru_numar_tranzactie NUMBER,
parametru_debitor NUMBER, parametru_creditor NUMBER, parametru_suma
NUMBER, parametru_descriere VARCHAR) AS
data_tranzactie DATE;
BEGIN
SELECT sysdate INTO data_tranzactie FROM dual;
INSERT INTO tranzactii VALUES (parametru_numar_tranzactie,
data_tranzactie, parametru_debitor, parametru_creditor, parametru_suma,
parametru_descriere);
END;
/
EXEC tranzactie (666, 22520, 42266, 8000, 'iPhone13');
select*from tranzactii;
--DELETE FROM tranzactii WHERE numar_tranzactie=666;
```

Implementare:

```
Run SQL Command Line

SQL> EXEC tranzactie (666, 22520, 42266, 8000, 'iPhone13');

PL/SQL procedure successfully completed.

Commit complete.
SQL> select*from tranzactii;

NUMAR_TRANZACTIE DATA_TRANZ NUMAR_CONT_DEBITOR NUMAR_CONT_CREDITOR SUMA_TRANZACTIONATA DESCRIERE_
---------------- ---------- ------------------ ------------------- ------------------- ----------
             111 04-06-2018              42266               22520                8805 InelAur
             222 22-05-2019              22520               69420                9688 Porsche911
             333 16-12-2020              69420               18069                3336 PlataClub
             444 18-09-2021              18069               33370                8187 Investitii
             555 03-03-2022              33370               42266                1989 Chirie
             666 27-05-2022              22520               42266                8000 iPhone13

6 rows selected.
```

Explicatii:

Se pot remarca in cadrul definitiei unei proceduri stocate doua componente primare:

- sectiunea de specificare a numelui procedurii si, optional, a parametrilor ei de apel;
- corpul procedurii, care contine comenzi SQL si/sau PL/SQL asigurand operatiile dorite.

Am creat, astfel, o procedura stocata ce implementeaza o tranzactie, folosind:

- parametru_numar_tranzactie=numar_tranzactie,
- parametru_debitor=numar_cont_debitor,
- parametru_creditor=numar_cont_creditor,
- parametru_suma=suma_tranzactionata,
- parametru_descriere=descriere_tranzactionata.

Toti acesti parametrii au fost introdusi intr-o variabila numita data_tranzactie, alaturi de data curenta din sistem, executand un nou articol in tabela ‘tranzactii’.

Caracterul ,/’ de la finalul procedurii conduce la compilarea procedurii.

Cerinta 4:

Sa se implementeze un trigger care atunci cand este introdusa o tranzactie, automat sa recalculeze soldul conturilor implicate in tranzactie.

Cod SQL:

```
CREATE OR REPLACE TRIGGER update_sold -- nu are parametrii
--deoarece triggerul se declanseaza pe o tabela si actioneaza pe alta,
este de tip AFTER
AFTER INSERT ON tranzactii
FOR EACH ROW -- deoarece folosim operatorii NEW si OLD
DECLARE -- declaram variabilele
Ex EXCEPTION;
Var_sold number (38,2);
BEGIN
SELECT sold INTO var_sold FROM Conturi WHERE numar_cont=
:NEW.numar_cont_debitor;
If var_sold>= :NEW.suma_tranzactionata THEN
UPDATE conturi SET sold = (sold - :NEW.suma_tranzactionata) WHERE
numar_cont =     :NEW.numar_cont_debitor;
Else
RAISE ex; -- se face rollback
END IF;
UPDATE conturi SET sold = (sold + :NEW.suma_tranzactionata) WHERE
numar_cont = :NEW.numar_cont_creditor;
EXCEPTION
WHEN ex THEN
RAISE_APPLICATION_ERROR (-20000, 'Sold insuficient');
WHEN OTHERS THEN
RAISE_APPLICATION_ERROR (-20000, 'EROARE');
END;
/

INSERT INTO tranzactii VALUES (777, '27.05.2022', 18069, 33370, 1000,
'Laptop'); --error, sold insuficient
INSERT INTO tranzactii VALUES (888, '27.05.2022', 33370, 42266, 2100,
'Laptop');
```

Implementare:

```
Trigger created.

SQL> INSERT INTO tranzactii VALUES (777, '27.05.2022', 18069, 33370, 1000, 'Laptop');
INSERT INTO tranzactii VALUES (777, '27.05.2022', 18069, 33370, 1000, 'Laptop')
            *
ERROR at line 1:
ORA-20000: Sold insuficient
ORA-06512: at "SYSTEM.UPDATE_SOLD", line 14
ORA-04088: error during execution of trigger 'SYSTEM.UPDATE_SOLD'
```

```
SQL> INSERT INTO tranzactii VALUES (888, '27.05.2022', 33370, 42266, 2100, 'Laptop');

1 row created.

SQL> select*from tranzactii;

NUMAR_TRANZACTIE DATA_TRANZ NUMAR_CONT_DEBITOR NUMAR_CONT_CREDITOR SUMA_TRANZACTIONATA DESCRIERE_
---------------- ---------- ------------------ ------------------- ------------------- ----------
             111 04-06-2018              42266               22520                8805 InelAur
             222 22-05-2019              22520               69420                9688 Porsche911
             333 16-12-2020              69420               18069                3336 PlataClub
             444 18-09-2021              18069               33370                8187 Investitii
             555 03-03-2022              33370               42266                1989 Chirie
             888 27-05-2022              33370               42266                2100 Laptop
             666 27-05-2022              22520               42266                8000 iPhone13

7 rows selected.

SQL> select*from conturi;

NUMAR_CONT DESCRIERE                                          TI SOLD_INITIAL       SOLD
---------- -------------------------------------------------- -- ------------ ----------
     42266 Irina Deaconescu                                   ac        14226      22042
     22520 Godeanu Lucian                                     ca        22520          0
     69420 Mirela Ionela                                      ac         6942      10565
     18069 Brailoiu Razvan                                    pa        34226          0
     33370 Cristina Maria                                     ca         3700       1600

SQL>
```

Explicatii:

Trigger-ul reprezinta o categorie speciala de proceduri stocate, fiind asociat unei tabele, declansandu-se automat (apelandu-se implicit) ca urmare a executiei unei comenzi SQL (INSERT, DELETE, UPDATE) pe tabela asociata. Spre deosebire de procedurile stocate, trigger-ele nu permit parametri de intrare.

Constructia CREATE TRIGGER poate fi definita pentru operatii de tip UPDATE, INSERT sau DELETE. Astfel, in definitia unui trigger, se pot folosi constructii sintactice de genul:

- BEFORE INSERT ON nume_tabela
- AFTER INSERT OR UPDATE OR DELETE ON nume_tabela

Eu am folosit constructia sintactica: AFTER INSERT ON tranzactii.

Se pot distinge doua categorii de triggere:

- BEFORE – inainte de aparitia evenimentului (mai precis, inaintea executiei comenzii DML care a declansat trigger-ul)
- AFTER - dupa aparitia evenimentului (mai precis dupa executia comenzii DML care a declansat trigger-ul).

Clauza FOR EACH ROW (definita ca optionala, dar obligatorie in anumite situatii) precizeaza daca trigger-ul se declanseaza o singura data pentru un eveniment (comanda DML) sau se declanseaza pentru fiecare linie afectata de eveniment.

Dupa cum se poate vedea in pozele de mai sus, dupa introducerea tranzactiei cu numarul ‘888’ din tabela conturi, s-au scazut 2100 din contul ‘33370’ (Cristina Maria) si s-au adaugat in contul ‘42266’ (Irina Deaconescu).

De asemenea, am folosit clauza UPDATE pentru a calcula soldurile conturilor implicate in tranzactie. Pentru contul debitor (din care pleaca banii) am aplicat scadere: sold = (sold - :NEW.suma_tranzactionata) , iar pentru contul creditor (in care intra banii) am aplicat adunare:
(sold + :NEW.suma_tranzactionata)

Am generat si o exceptie, punand conditia:

If var_sold>=:NEW.suma_tranzactionata THEN, astfel ca daca nu exista bani in contul debitorului, se va afiseaza mesajul “'Sold insuficient”.

Cerinta 5:

Să se creeze un raport in care sa se afişeze toate tranzacţiile în care este folosit un anumit cont.

Cod SQL:

```
SELECT * FROM tranzactii WHERE numar_cont_debitor=22520 OR
numar_cont_creditor = 22520;
```

Implementare:

```
SQL> SELECT * FROM tranzactii WHERE numar_cont_debitor=22520 OR numar_cont_creditor = 22520;

NUMAR_TRANZACTIE DATA_TRANZ NUMAR_CONT_DEBITOR NUMAR_CONT_CREDITOR SUMA_TRANZACTIONATA DESCRIERE_
---------------- ---------- ------------------ ------------------- -------------------- ----------
             111 04-06-2018              42266               22520                 8805 InelAur
             222 22-05-2019              22520               69420                 9688 Porsche911
             666 27-05-2022              22520               42266                 8000 iPhone13
```

Explicatii:

Pentru afisarea tuturor tranzactiilor folosim clauza: SELECT * FROM [nume_tabela].

Pentru afisarea tuturor tranzactiilor a unui singur cont folosim clauza: WHERE [conditie].

Mai sus, putem observa ca am afisat tranzactiile ale unui cont (Godeanu Lucian-22520), care este atat creditor, cat si debitor.

Cerinta 6:

Să se afişeze toate tranzactiile care au fost introduse in perioada (01.01.2020- 01.06.2020).

Cod SQL:

```
SELECT * FROM tranzactii WHERE data_tranzactie >= '01.01.2020' AND
data_tranzactie <= '01.06.2020'; --error
SELECT * FROM tranzactii WHERE data_tranzactie >= '03.03.2022' AND
data_tranzactie <= '27.05.2022';
```

Implementare:

```
SQL> SELECT * FROM tranzactii WHERE data_tranzactie >= '01.01.2020' AND data_tranzactie <= '01.06.2020';

no rows selected

SQL> SELECT * FROM tranzactii WHERE data_tranzactie >= '03.03.2022' AND data_tranzactie <= '27.05.2022';

NUMAR_TRANZACTIE DATA_TRANZ NUMAR_CONT_DEBITOR NUMAR_CONT_CREDITOR SUMA_TRANZACTIONATA DESCRIERE_
---------------- ---------- ------------------ ------------------- -------------------- ----------
             555 03-03-2022              33370               42266                 1989 Chirie
             888 27-05-2022              33370               42266                 2100 Laptop
```

Explicatii:

Pentru afisarea tuturor tranzactiilor folosim clauza: SELECT * FROM [nume_tabela].

Pentru afisarea tuturor tranzactiilor a unui singur cont folosim clauza: WHERE [conditie].

Unele criterii sunt simple si utilizeaza operatori si constante de baza. Altele sunt complexe si utilizeaza functii si operatori speciali si includ referinte de camp.

Mai mult decat atat, clauza WHERE se poate folosi cu mai multe conditii (01.01.2020 – 01.06.2020). Datorita faptului ca eu nu am introdus decat un articol cu data ’16.12.2020’, nu se poate afisa nimic.

Pentru a arata ca totusi functioneaza, am schimbat data intre 03.03.2020 – 27.05.2020.

Cerinta 7:

Să se calculeze şi să se afişeze suma totala creditoare, respectiv debitoare, pentru toate conturile care au fost implicate in tranzactii.

Cod SQL:

```
--suma totala debitoare
CREATE VIEW total_debit AS
SELECT A.numar_cont, B.numar_cont_debitor, SUM(B.suma_tranzactionata) AS
suma_debitoare
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_debitor(+)
GROUP BY A.numar_cont, B.numar_cont_debitor
ORDER BY numar_cont;
SELECT * FROM total_debit;
SELECT SUM (suma_debitoare) FROM total_debit;

--suma totala creditoare
CREATE VIEW total_credit AS
SELECT A.numar_cont, B.numar_cont_creditor, SUM(B.suma_tranzactionata) AS
suma_creditoare
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_creditor(+)
GROUP BY A.numar_cont, B.numar_cont_creditor
ORDER BY numar_cont;
SELECT * FROM total_credit;
SELECT SUM (suma_creditoare) FROM total_credit;
```

Implementare:

```
View created.

SQL> SELECT * FROM total_debit;

NUMAR_CONT NUMAR_CONT_DEBITOR SUMA_DEBITOARE
---------- ------------------ --------------
     18069              18069           8187
     22520              22520          17688
     33370              33370           4089
     42266              42266           8805
     69420              69420           3336

SQL> SELECT SUM (suma_debitoare) FROM total_debit;

SUM(SUMA_DEBITOARE)
-------------------
              42105
```

```
View created.

SQL> SELECT * FROM total_credit;

NUMAR_CONT NUMAR_CONT_CREDITOR SUMA_CREDITOARE
---------- ------------------- ---------------
     18069               18069            3336
     22520               22520            8805
     33370               33370            8187
     42266               42266           12089
     69420               69420            9688

SQL> SELECT SUM (suma_creditoare) FROM total_credit;

SUM(SUMA_CREDITOARE)
--------------------
               42105
```

Explicatii:

La acest subpunct, m-am gandit sa creez 2 tabele virtuale numite ‘total_debit’ si ‘total_credit’ cu ajutorul carora am calculat suma debitoare, respectiv suma creditoare pentru fiecare cont in parte, folosind clauza WHERE:

- WHERE A.numar_cont=B.numar_cont_debitor(+)
- WHERE A.numar_cont=B.numar_cont_creditor(+)

Semnul (+) de la sfarsitul clauzei WHERE se refera la faptul ca, daca unei linii din prima tabela referita in clauza FROM (in exemplul nostru, tabela ‘conturi’) nu îi corespund linii in a doua tabela din lista (in exemplul nostru, tabela ‘tranzactii’), aceste linii nu sunt afisate.

Astfel, se poate observa ca sunt afisate inclusiv liniile din prima tabela pentru care nu exista corespondent in cealalta tabela (datele lipsa primind valori NULL).

In cele din urma, pentru a afisa suma totala debitoare/creditoare am folosit clauzele SUM si SELECT:

- SELECT SUM (suma_debitoare) FROM total_debit;
- SELECT SUM (suma_creditoare) FROM total_credit;

Cerinta 8:

Să se afişeze toate tranzacţiile care implică conturi de un anumit tip (’ca’ –capital, ’pa’-pasiv, sau ’ac’-activ).

Cod SQL:

```
--A. pentru tip_cont='ca':
--1. nr_cont_debitor
SELECT A.tip_cont, A.numar_cont, B.numar_tranzactie,
B.suma_tranzactionata, B.numar_cont_debitor, B.descriere_tranzactie
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_debitor AND A.tip_cont='ca'
ORDER BY numar_cont;

--2. nr_cont_creditor
SELECT A.tip_cont, A.numar_cont, B.numar_tranzactie,
B.suma_tranzactionata, B.numar_cont_creditor, B.descriere_tranzactie
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_creditor AND A.tip_cont='ca'
ORDER BY numar_cont;

--B. pentru tip_cont='pa';
--1. nr_cont_debitor
SELECT A.tip_cont, A.numar_cont, B.numar_tranzactie,
B.suma_tranzactionata, B.numar_cont_debitor, B.descriere_tranzactie
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_debitor AND A.tip_cont='pa'
ORDER BY numar_cont;

--2. nr_cont_creditor
SELECT A.tip_cont, A.numar_cont, B.numar_tranzactie,
B.suma_tranzactionata, B.numar_cont_creditor, B.descriere_tranzactie
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_creditor AND A.tip_cont='pa';

--C. tip_cont='ac';
--1. nr_cont_debitor
SELECT A.tip_cont, A.numar_cont, B.numar_tranzactie,
B.suma_tranzactionata, B.numar_cont_debitor, B.descriere_tranzactie
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_debitor AND A.tip_cont='ac'
ORDER BY numar_cont;

--2. nr_cont_creditor
SELECT A.tip_cont, A.numar_cont, B.numar_tranzactie,
B.suma_tranzactionata, B.numar_cont_creditor, B.descriere_tranzactie
FROM conturi A, tranzactii B
WHERE A.numar_cont=B.numar_cont_debitor AND A.tip_cont='ac'
ORDER BY numar_cont;
```

Implementare:

```
TI NUMAR_CONT NUMAR_TRANZACTIE SUMA_TRANZACTIONATA NUMAR_CONT_DEBITOR DESCRIERE_
-- ---------- ---------------- ------------------ ------------------ ----------
ca      22520              222               9688              22520 Porsche911
ca      22520              666               8000              22520 iPhone13
ca      33370              888               2100              33370 Laptop
ca      33370              555               1989              33370 Chirie
```

(Pentru tip_cont='ca': nr_cont_debitor)

```
TI NUMAR_CONT NUMAR_TRANZACTIE SUMA_TRANZACTIONATA NUMAR_CONT_CREDITOR DESCRIERE_
-- ---------- ---------------- ------------------ ------------------- ----------
ca      22520              111               8805               22520 InelAur
ca      33370              444               8187               33370 Investitii
```

(Pentru tip_cont='ca': nr_cont_creditor)

```
TI NUMAR_CONT NUMAR_TRANZACTIE SUMA_TRANZACTIONATA NUMAR_CONT_DEBITOR DESCRIERE_
-- ---------- ---------------- ------------------ ------------------ ----------
pa      18069              444               8187              18069 Investitii
```

(Pentru tip_cont='pa': nr_cont_debitor)

```
TI NUMAR_CONT NUMAR_TRANZACTIE SUMA_TRANZACTIONATA NUMAR_CONT_CREDITOR DESCRIERE_
-- ---------- ---------------- ------------------ ------------------- ----------
pa      18069              333               3336               18069 PlataClub
```

(Pentru tip_cont='pa': nr_cont_creditor)

```
TI NUMAR_CONT NUMAR_TRANZACTIE SUMA_TRANZACTIONATA NUMAR_CONT_DEBITOR DESCRIERE_
-- ---------- ---------------- ------------------ ------------------ ----------
ac      42266              111               8805              42266 InelAur
ac      69420              333               3336              69420 PlataClub
```

(Pentru tip_cont='ac': nr_cont_debitor)

```
TI NUMAR_CONT NUMAR_TRANZACTIE SUMA_TRANZACTIONATA NUMAR_CONT_CREDITOR DESCRIERE_
-- ---------- ---------------- ------------------ ------------------- ----------
ac      42266              111               8805               22520 InelAur
ac      69420              333               3336               18069 PlataClub
```

(Pentru tip_cont='ac': nr_cont_creditor)

Explicatii:

Enuntul acestui subpunct ne cere afisarea tuturor tranzactiilor care implica conturi de un anumit tip. Astfel, am luat toate posibilitatile, pentru a afisa toate tranzactiile.

Din acest punct de vedere, am folosit relationarea celor doua tabele (A-conturi, B-tranzactii) pentru a realiza programele, folosind clauza WHERE unde am impus urmatoarele conditii:

Pentru ac:

- A.numar_cont=B.numar_cont_debitor AND A.tip_cont='ac'
- A.numar_cont=B.numar_cont_creditor AND A.tip_cont='ac'

Pentru pa:

- A.numar_cont=B.numar_cont_debitor AND A.tip_cont='pa'
- A.numar_cont=B.numar_cont_creditor AND A.tip_cont='pa'

Pentru ca:

- A.numar_cont=B.numar_cont_debitor AND A.tip_cont='ca'
- A.numar_cont=B.numar_cont_creditor AND A.tip_cont='ca’

Am aplicat, in acest scop, 6 programe pentru a identifica toate tranzactiile in functie de tipul contului.

Cerinta 9:

Să se steargă un cont daca nu exista tranzactii pentru el.

Cod SQL:

```
DELETE FROM conturi
WHERE NOT EXISTS (
SELECT numar_cont_debitor, numar_cont_creditor FROM tranzactii );
--sau
DELETE FROM conturi WHERE numar_cont NOT IN (
SELECT numar_cont_debitor FROM tranzactii) AND numar_cont NOT IN (
SELECT numar_cont_creditor FROM tranzactii );
```

Implementare:

```
SQL> DELETE FROM conturi
  2  WHERE NOT EXISTS (
  3  SELECT numar_cont_debitor, numar_cont_creditor FROM tranzactii );

0 rows deleted.

SQL> DELETE FROM conturi WHERE numar_cont NOT IN (
  2  SELECT numar_cont_debitor FROM tranzactii) AND numar_cont NOT IN (
  3  SELECT numar_cont_creditor FROM tranzactii );

0 rows deleted.
```

Explicatii:

Operatorul NOT EXISTS funcționeaza opus operatorului EXISTS.

Folosim adesea operatorul NOT EXISTS cu o subinterogare pentru a scadea un set de date din altul.

De asemenea, am folosit si clauza NOT IN echivalent cu !=ALL. Evaluează la FALS dacă vreun membru al setului este NULL.

Deoarece am folosit toate conturile introduse la inceput, atat pentru debitor cat si pentru creditor, functia nu imi va sterge niciun articol din tabela ’conturi’.

Cerinta 10:

Să se afiseze contul care apare in cele mai multe tranzactii precum si numarul de tranzactii in care el apare.

Cod SQL:

```
CREATE VIEW Debitor AS(
SELECT Numar_cont_debitor, count(Numar_cont_debitor) as nr_aparitii
FROM Tranzactii
GROUP BY numar_cont_debitor);
select*from Debitor;

CREATE VIEW Creditor AS(
SELECT Numar_cont_creditor, count(Numar_cont_creditor) AS nr_aparitii_cred
FROM Tranzactii
GROUP BY numar_cont_creditor);
select*from Creditor;

SELECT numar_cont, sum(nr_aparitii+nr_aparitii_cred) AS maxim_aparitii
FROM Conturi A, Debitor B, Creditor C
WHERE A.numar_cont=B.Numar_cont_debitor(+) AND
A.numar_cont=C.numar_cont_creditor(+)
HAVING sum(nr_aparitii+nr_aparitii_cred) = (SELECT
max(sum(nr_aparitii+nr_aparitii_cred))
FROM Conturi A, Debitor B, Creditor C
WHERE A.numar_cont=B.Numar_cont_debitor(+) AND
A.numar_cont=C.numar_cont_creditor(+)
GROUP BY numar_cont)
GROUP BY numar_cont;
```

Implementare:

```
View created.

SQL> select*from Debitor;

NUMAR_CONT_DEBITOR NR_APARITII
------------------ -----------
             42266           1
             69420           1
             18069           1
             33370           2
             22520           2
```

```
View created.

SQL> select*from Creditor;

NUMAR_CONT_CREDITOR NR_APARITII_CRED
------------------- ----------------
              42266                3
              69420                1
              18069                1
              33370                1
              22520                1
```

```
SQL> SELECT numar_cont, sum(nr_aparitii+nr_aparitii_cred) AS maxim_aparitii
  2  FROM Conturi A, Debitor B, Creditor C
  3  WHERE A.numar_cont=B.Numar_cont_debitor(+) AND A.numar_cont=C.numar_cont_creditor(+)
  4  HAVING sum(nr_aparitii+nr_aparitii_cred) = (SELECT max(sum(nr_aparitii+nr_aparitii_cred))
  5  FROM Conturi A, Debitor B, Creditor C
  6  WHERE A.numar_cont=B.Numar_cont_debitor(+) AND A.numar_cont=C.numar_cont_creditor(+)
  7  GROUP BY numar_cont)
  8  GROUP BY numar_cont;

NUMAR_CONT MAXIM_APARITII
---------- --------------
     42266              4

SQL>
```

Explicatii:

Pentru acest subpunct am creat alte 2 tabele virtuale numite ‘debit’ si ‘credit’ cu ajutorul carora ne afiseaza numarul fiecarui cont dar si numarul de aparitii ale acestora.

Nu in ultimul rand, am folosit clauza HAVING alaturi de sum(nr_aparitii+nr_aparitii_cred)=(SELECTmax(sum(nr_aparitii+nr_aparitii_cred)) pentru a afla numarul maxim de aparitiidin cele doua tabele.

Bibliografie:

1. Curs PL/SQL Baze de Date UPT
2. Anexe PL/SQL Baze de Date UPT
3. Laboratoare PL/SQL Baze de Date UPT
4. W3schools.com – SQL Course