

Alexis
PLESSIER
11/01/2021

N3 Informatique

Implémentation des SGBD Rendu TP2

Exercice 1 :

```
CREATE TABLE COMPTE_CHEQUE
(NUM_CC number(4) CONSTRAINT PK_COMPTE_CHEQUE PRIMARY KEY,
 NUM_CLIENT number(4),
 SOLDE number(7,2));

CREATE TABLE COMPTE_EPARGNE
(NUM_CE number(4) CONSTRAINT PK_COMPTE_EPARGNE PRIMARY KEY,
 NUME_CLIENT number(4),
 SOLDE number(7,2));

CREATE TABLE DECISION_CREDIT
(NUM_DOSSIER number(4) CONSTRAINT PK_DECISION_CREDIT PRIMARY KEY,
 NUM_CLIENT number(4),
 DECISION char(2));

INSERT INTO COMPTE_CHEQUE VALUES(0001,8888,9000);
INSERT INTO COMPTE_CHEQUE VALUES(0002,1111,2000);
INSERT INTO COMPTE_CHEQUE VALUES(1234,1234,800);
INSERT INTO COMPTE_CHEQUE VALUES(9999,9000,5000);

INSERT INTO COMPTE_EPARGNE VALUES(0001,8888,1000);
INSERT INTO COMPTE_EPARGNE VALUES(0000,9000,100);
INSERT INTO COMPTE_EPARGNE VALUES(1111,1111,8000);
INSERT INTO COMPTE_EPARGNE VALUES(1234,1234,9000);
```

Exercice 2 :

(a)

Solde suffisant :

```
SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT  SOLDE
-----
      1      8888    9000
      2     1111    2000
    1234    1234      800
    9999     9000    5000

SQL> exec transfert(8888,0001,0002,1000);
Operation effectuee !

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT  SOLDE
-----
      1      8888    7000
      2     1111    4000
    1234    1234      800
    9999     9000    5000

Create or replace procedure transfert(num_c number, frm_num_cpt number,
t_num_cpt number, m number) AS
s number;
n number;

Begin
    SELECT SOLDE INTO s FROM COMPTE_CHEQUE WHERE NUM_CC = frm_num_cpt;
    SELECT NUM_CC INTO n FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;

    IF(n != frm_num_cpt) THEN
        Dbms_Output.Put_Line('Numero Client et Numero Compte non associés');

    ELSIF(s-m < 0) THEN
        Dbms_Output.Put_Line('Solde Insuffisant pour le client no : '||
num_c);
    ELSE
        UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE + m WHERE NUM_CC = t_num_cpt;
        UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE - m WHERE NUM_CC =
frm_num_cp t;
        Dbms_Output.Put_Line('Operation effectuee !');

    END IF;

End;
```

Le client numéro 8888 a bien virer 1000 unités de son compte (0001) au compte 0002 du client numéro 1111.

Solde insuffisant :

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	800
9999	9000	5000

```
SQL> exec transfert(8888,0001,0002,8000);
Solde Insuffisant pour le client no : 8888

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	800
9999	9000	5000

Erreur Numéro Client/Compte :

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	800
9999	9000	5000

```
SQL> exec transfert(1111,0001,0002,8000);
Numero Client et Numero Compte non associes

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	800
9999	9000	5000

(b)

Solde suffisant :

Le client numéro 1234 a bien viré 300 unités de son compte épargne à son compte chèque.

```
SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUM_CLIENT	SOLDE
1	8888	1000
0	9000	100
1111	1111	8000
1234	1234	9000

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	800
9999	9000	5000

```
SQL> virement_cc(1234,1234,300);
SP2-0734: unknown command beginning "virement_c..." - rest of line ignored.
SQL> exec virement_cc(1234,1234,300);
Operation effectuee !

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUM_CLIENT	SOLDE
1	8888	1000
0	9000	100
1111	1111	8000
1234	1234	8700

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	1100
9999	9000	5000

```
SQL>
```

```
Create or replace procedure virement_cc(num_c number, numcc number, m
number) AS
s number;
n number;
Begin
SELECT SOLDE INTO s FROM COMPTE_EPARGNE WHERE NUM_CLIENT = num_c;
SELECT NUM_CC INTO n FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;

IF(n != numcc) THEN
Dbms_Output.Put_Line('Numero Client et Numero Compte non associes');

ELSIF(s-m < 0) THEN
Dbms_Output.Put_Line('Solde Insuffisant pour le client no : '||
num_c);
ELSE
UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE + m WHERE NUM_CC = numcc;
UPDATE COMPTE_EPARGNE SET SOLDE = SOLDE - m WHERE NUM_CLIENT =
num_c;
Dbms_Output.Put_Line('Operation effectuee !');

END IF;
End;
```

Solde insuffisant :

```
SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUME_CLIENT	SOLDE
1	8888	1000
0	9000	100
1111	1111	8000
1234	1234	8700

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	1100
9999	9000	5000

```
SQL> exec virement_cc(1234,1234,9000);
Solde Insuffisant pour le client no : 1234

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUME_CLIENT	SOLDE
1	8888	1000
0	9000	100
1111	1111	8000
1234	1234	8700

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	1100
9999	9000	5000

```
SQL>
```

Erreur Numéro Client/Compte :

```
SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUME_CLIENT	SOLDE
1	8888	1000
0	9000	100
1111	1111	8000
1234	1234	8700

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	1100
9999	9000	5000

```
SQL> exec virement_cc(1234,0001,9000);
Numero Client et Numero Compte non associes

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUME_CLIENT	SOLDE
1	8888	1000
0	9000	100
1111	1111	8000
1234	1234	8700

```
SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7000
2	1111	4000
1234	1234	1100
9999	9000	5000

```
SQL>
```

(c)

Solde suffisant :

Le client numéro 8888 a bien viré 5000 unités de son compte chèque à son compte épargne.

```
SQL> exec virement_cc(8888,0001,5000);
Operation effectuee !

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	7100
2	1111	4000
1234	1234	1100
9999	9000	5000

```
SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUME_CLIENT	SOLDE
1	8888	900
0	9000	100
1111	1111	8000
1234	1234	8700

```
SQL> exec virement_cc(8888,0001,5000);
Operation effectuee !

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;
```

NUM_CC	NUM_CLIENT	SOLDE
1	8888	2100
2	1111	4000
1234	1234	1100
9999	9000	5000

```
SQL> select * from COMPTE_EPARGNE;
```

NUM_CE	NUME_CLIENT	SOLDE
1	8888	5900
0	9000	100
1111	1111	8000
1234	1234	8700

```
SQL>
```

```

--creation-BD.sql
Create or replace procedure virement_cc(num_c number, numce number, m
number) AS
s number;
n number;

Begin

SELECT SOLDE INTO s FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;
SELECT NUM_CE INTO n FROM COMPTE_EPARGNE WHERE NUME_CLIENT = num_c;

IF(n != numce) THEN
    Dbms_Output.Put_Line('Numero Client et Numero Compte non associes');

ELSIF(s-m < 0) THEN
    Dbms_Output.Put_Line('Solde Insuffisant pour le client no : '||
num_c);
ELSE
    UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE - m WHERE NUM_CLIENT = num_c;
    UPDATE COMPTE_EPARGNE SET SOLDE = SOLDE + m WHERE NUME_CLIENT =
num_c;
    Dbms_Output.Put_Line('Operation effectuee !');

END IF;

End;
```

Solde insuffisant :

```
SQL> exec virement_ce(8888,0001,5000);
Solde Insuffisant pour le client no : 8888

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT    SOLDE
-----
      1      8888      2100
      2      1111      4000
    1234     1234      1100
    9999     9000      5000

SQL> select * from COMPTE_EPARGNE;

  NUM_CE NUME_CLIENT    SOLDE
-----
      1      8888      5900
      0      9000       100
    1111     1111      8000
    1234     1234      8700
```

Erreur Numéro Client/Compte :

```
SQL> exec virement_ce(8888,0002,5000);
Numero Client et Numero Compte non associes

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_EPARGNE;

  NUM_CE NUME_CLIENT    SOLDE
-----
      1      8888      5900
      0      9000       100
    1111     1111      8000
    1234     1234      8700

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT    SOLDE
-----
      1      8888      2100
      2      1111      4000
    1234     1234      1100
    9999     9000      5000

SQL>
```

(d)

Crédit accepter :

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> Select * from DECISION_CREDIT;

 NUM_DOSSIER NUM_CLIENT DE
-----
      1          0 TE

SQL> traitement_credit(8888,2000);
SP2-0734: unknown command beginning "traitement..." - rest of line ignored.
SQL> exec traitement_credit(8888,2000);

PL/SQL procedure successfully completed.

SQL> Select * from DECISION_CREDIT;

 NUM_DOSSIER NUM_CLIENT DE
-----
      1          0 TE
      2      8888 OK

SQL>
```

```
Create or replace procedure traitement_credit(num_c number, m number) AS
s1 number;
s2 number;
calc_index number;

Begin

  SELECT SOLDE INTO s1 FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;
  SELECT SOLDE INTO s2 FROM COMPTE_EPARGNE WHERE NUME_CLIENT = num_c;
  SELECT max(NUM_DOSSIER) INTO calc_index FROM DECISION_CREDIT;

  IF(s1+s2 < (1/3)*m) THEN
    INSERT INTO DECISION_CREDIT VALUES(calc_index+1,num_c,'KO');
  ELSE
    INSERT INTO DECISION_CREDIT VALUES(calc_index+1,num_c,'OK');
  END IF;
End;
```

Crédit refuser :

```
SQL> exec traitement_credit(8888,25000);

PL/SQL procedure successfully completed.

SQL> Select * from DECISION_CREDIT;

 NUM_DOSSIER NUM_CLIENT DE
-----
      1          0 TE
      2      8888 OK
      3      8888 OK
      4      8888 KO

SQL>
```

Exercice 3 :

- (a) Ajout de *dbms_lock.sleep(nb_seconds IN NUMBER)* dans les procédures. Pour une question pratique, toutes les procédures se sont vu attribuer un paramètre en plus qui est « **sec number** ». Cette variable est ensuite appliquée à la fonction DBMS_LOCK.sleep(sec) dans les procédures.

(b)

Transfert d'argent de deux personnes différentes sur un même compte :

Deux comptes envoient 1000 unités sur le compte numéro 0001 qui possède un solde de 4100. Sans problème de concurrence, le solde devrait être de 6100.

Session1

```
SQL> SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT  SOLDE
-----
      1      8888      4100
      2      1111      3000
    1234     1234      1100
    9999     9000      4000

SQL> exec transfert(9000,9999,0001,1000,5);

PL/SQL procedure successfully completed.

SQL> commit;

Commit complete.

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT  SOLDE
-----
      1      8888      5100
      2      1111      3000
    1234     1234      1100
    9999     9000      3000

SQL> _
```

Session 2

```
SQL> exec transfert(1111,0002,0001,1000,0);

PL/SQL procedure successfully completed.

SQL> commit;

Commit complete.

SQL> exec transfert(1111,0002,0001,1000,4);

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL>
SQL>
SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT  SOLDE
-----
      1      8888      6100
      2      1111      2000
    1234     1234      1100
    9999     9000      3000

SQL> □
```

On remarque que la session 1 possède une base de données erronée.

Transfert d'argent (d'un tiers) sur le compte chèque ET virement du compte chèque au compte épargne :

Session 1

```

SQL> select * from COMPTE_EPARGNE;

  NUM_CE NUME_CLIENT SOLDE
-----
      1      8888      6900
      0      9000       100
     1111      1111      8000
     1234      1234      8700

SQL> transfert(1234,1234,0001,200,5);
SP2-0734: unknown command beginning "transfert(..." - rest of line ignored.
SQL> exec transfert(1234,1234,0001,200,5);

PL/SQL procedure successfully completed.

SQL> commit;

Commit complete.

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT SOLDE
-----
      1      8888      7500
      2      1111      1000
     1234      1234       500
     9999      9000      2200

```

Session 2

```

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT SOLDE
-----
      1      8888      7300
      2      1111      1000
     1234      1234       700
     9999      9000      2200

SQL> select * from COMPTE_EPARGNE;

  NUM_CE NUME_CLIENT SOLDE
-----
      1      8888      6900
      0      9000       100
     1111      1111      8000
     1234      1234      8700

SQL> exec virement_ce(8888,0001,500,0);

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT SOLDE
-----
      1      8888      7000
      2      1111      1000
     1234      1234       500
     9999      9000      2200

```

Un compte tiers transfert de l'argent sur un compte chèque qui, en même temps, envoie de l'argent sur son compte épargne, problème de concurrence.

Transfert d'argent ET demande de crédit :

Session 1

```

SQL> select * from COMPTE_EPARGNE;

  NUM_CE NUME_CLIENT SOLDE
-----
      1      8888      7400
      0      9000       100
     1111      1111      8000
     1234      1234      8700

SQL> exec transfert(9000,9999,0001,2000,5);

PL/SQL procedure successfully completed.

SQL> select * from COMPTE_CHEQUE;

  NUM_CC NUM_CLIENT SOLDE
-----
      1      8888      9000
      2      1111      1000
     1234      1234       500
     9999      9000       200

SQL> select * from COMPTE_EPARGNE;

  NUM_CE NUME_CLIENT SOLDE
-----
      1      8888      7400
      0      9000       100

```

Session 2

```

SQL>
SQL>
SQL>
SQL>
SQL>
SQL> exec traitement_credit(9000,1000,0);

PL/SQL procedure successfully completed.

SQL> select * from TRAITEMENT_CREDIT;
select * from TRAITEMENT_CREDIT
*
ERROR at line 1:
ORA-04044: procedure, function, package, or type is not allowed here

SQL> select * from DECISION_CREDIT;

 NUM_DOSSIER NUM_CLIENT DE
-----
      5      1234 K0
      6      1234 K0
      7      9000 OK
      1         0 TE
      2      8888 OK
      3      8888 OK
      4      8888 K0

```


Le client numéro 9000 avait un solde tous compte de $2200+100 = 2300$ unités.

A ce moment il pouvait emprunter 1000 unités. Il a ensuite fait un virement de 2000 unité au compte chèque numéro 0001, son solde tous compte est donc passé à $200+100 = 300$ unités. Il à demandé à emprunté 1000 unités et comme le montre la 3^{ème} ligne de la table DECISION_CREDIT, ça lui a été accepté alors qu'il possède moins de 3x la somme qu'il souhaite emprunter, problème de concurrence.

Exercice 4 :

Sous Oracle, les instructions entrainent déjà des verrous :

Commande SQL	Mode de Verrou
SELECT ... FROM table ...	Aucun verrou
INSERT INTO table ...	RX
UPDATE TABLE ...	RX
UPDATE table ...	RX
DELETE FROM ...	RX

Où : **RX** = Row Exclusive

SRX = Share Row Exclusive

X = Exclusive (Mode le plus restrictif)

Cependant, j'ai quand même modifié le verrou des changements les plus sensibles (**UPDATE**) en verrou Exclusif (**LOCK TABLE nom_table IN EXCLUSIVE MODE**).

Exercice 5 :

Il suffit d'ajouter un **COMMIT** à la fin des procédures pour éviter ces problèmes de concurrence.

Code final :

transfert :

```
Create or replace procedure transfert(num_c number, frm_num_cpt number, t_num_cpt number, m number, sec number) AS
s number;
n number;

Begin

    SELECT SOLDE INTO s FROM COMPTE_CHEQUE WHERE NUM_CC = frm_num_cpt;
    SELECT NUM_CC INTO n FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;

    IF(n != frm_num_cpt) THEN
        Dbms_Output.Put_Line('Numero Client et Numero Compte non associes');
    ELSIF(s-m < 0) THEN
        Dbms_Output.Put_Line('Solde Insuffisant pour le client no : '|| num_c);
    ELSE

        UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE + m WHERE NUM_CC = t_num_cpt;
        LOCK TABLE COMPTE_CHEQUE IN EXCLUSIVE MODE;
        DBMS_LOCK.sleep(sec);
        UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE - m WHERE NUM_CC = frm_num_cpt;
        LOCK TABLE COMPTE_CHEQUE IN EXCLUSIVE MODE;
        Dbms_Output.Put_Line('Operation effectuee !');

    END IF;

End;
```

virement_cc :

```
Create or replace procedure virement_cc(num_c number, numcc number, m number, sec number) AS
s number;
n number;

Begin

    SELECT SOLDE INTO s FROM COMPTE_EPARGNE WHERE NUM_CLIENT = num_c;
    SELECT NUM_CC INTO n FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;

    IF(n != numcc) THEN
        Dbms_Output.Put_Line('Numero Client et Numero Compte non associes');
    ELSIF(s-m < 0) THEN
        Dbms_Output.Put_Line('Solde Insuffisant pour le client no : '|| num_c);
    ELSE

        UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE + m WHERE NUM_CC = numcc;
        LOCK TABLE COMPTE_CHEQUE IN EXCLUSIVE MODE;
        DBMS_LOCK.sleep(sec);
        UPDATE COMPTE_EPARGNE SET SOLDE = SOLDE - m WHERE NUM_CLIENT = num_c;
        LOCK TABLE COMPTE_EPARGNE IN EXCLUSIVE MODE;
        Dbms_Output.Put_Line('Operation effectuee !');

    END IF;

End;
```

virement_ce :

```
Create or replace procedure virement_ce(num_c number, numce number, m number, sec number) AS

s number;
n number;

Begin

    SELECT SOLDE INTO s FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;
    SELECT NUM_CE INTO n FROM COMPTE_EPARGNE WHERE NUME_CLIENT = num_c;

    IF(n != numce) THEN
        Dbms_Output.Put_Line('Numero Client et Numero Compte non associes');
    ELSIF(s-m < 0) THEN
        Dbms_Output.Put_Line('Solde Insuffisant pour le client no : ' || num_c);
    ELSE

        UPDATE COMPTE_CHEQUE SET SOLDE = SOLDE - m WHERE NUM_CLIENT = num_c;
        LOCK TABLE COMPTE_CHEQUE IN EXCLUSIVE MODE;
        DBMS_LOCK.sleep(sec);
        UPDATE COMPTE_EPARGNE SET SOLDE = SOLDE + m WHERE NUME_CLIENT = num_c;
        LOCK TABLE COMPTE_EPARGNE IN EXCLUSIVE MODE;
        Dbms_Output.Put_Line('Operation effectuee !');

    END IF;
End;
```

traitement_credit :

```
Create or replace procedure traitement_credit(num_c number, m number, sec number) AS

s1 number;
s2 number;
calc_index number;

Begin

    SELECT SOLDE INTO s1 FROM COMPTE_CHEQUE WHERE NUM_CLIENT = num_c;
    SELECT SOLDE INTO s2 FROM COMPTE_EPARGNE WHERE NUME_CLIENT = num_c;
    SELECT max(NUM_DOSSIER) INTO calc_index FROM DECISION_CREDIT;

    DBMS_LOCK.sleep(sec);
    IF(s1+s2 < (1/3)*m) THEN
        INSERT INTO DECISION_CREDIT VALUES(calc_index+1,num_c,'KO');
    ELSE
        INSERT INTO DECISION_CREDIT VALUES(calc_index+1,num_c,'OK');
    END IF;

End;
```