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:

```
QL> select * from COMPTE_CHEQUE;
    NUM CC NUM CLIENT
                                      SOLDE
                                                                                                                     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;
QL> exec transfert(8888,0001,0002,1000);
                                                                                                                           IF(n != frm num_cpt) THEN
    Dbms_Output.Put_Line('Numero Client et Numero Compte non associés');
PL/SQL procedure successfully completed.
                                                                                                                           ELSIF(s-m < 0) THEN
    Dbms_Output.Put_Line('Solde Insuffisant pour le client no : '||</pre>
OL> select * from COMPTE CHEOUE:
                                                                                                                      num c);
    NUM CC NUM CLIENT
                                                                                                                      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 !');
       1
2
1234
9999
                                                                                                                           END IF;
```

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 COMP	TE_CHEQUE;	
NUM_CC NUM	_CLIENT	SOLDE	
1	8888	7000	
	1111		
1234	1234	800	
9999	9000	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	7000	
	1111		
	1234		
9999	9000	5000	

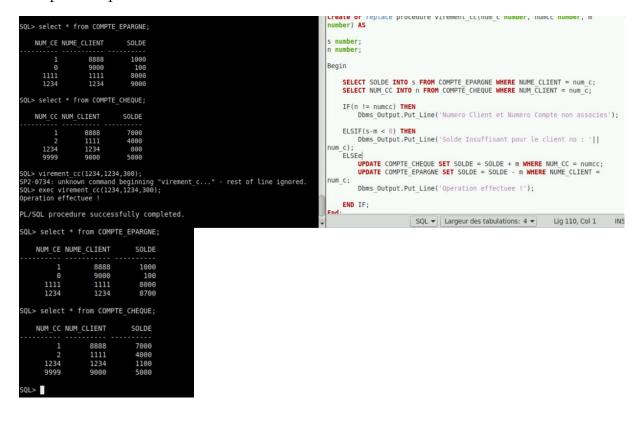
Erreur Numéro Client/Compte:

SQL> select *	from COMPT	E_CHEQUE;		
NUM_CC NUM_CLIENT SOLDE				
1	0000	7000		
	8888			
	1111			
1234				
9999	9000	5000		
SQL> exec trans				
Numero Client (et Numero	Compte non ass	ocies	
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.



Solde insuffisant:

SQL> select *	from COMPT	E_EPARGNE;	
NUM_CE NUM	ME_CLIENT	SOLDE	
1	0000	1000	
0	8888 9000	1000 100	
1111	1111	8000	
1234	1234		
1254	1234	8700	
SQL> select *	from COMPT	E_CHEQUE;	
NUM_CC NUM	_CLIENT	SOLDE	
1	8888	7000	
2	1111		
1234	1234	1100	
9999	1234 9000	5000	
SQL> exec virement_cc(1234,1234,9000); Solde Insuffisant pour le client no : 1234 PL/SQL procedure successfully completed.			
PL/SQL procedu	ire success	rully completed.	
SQL> select * from COMPTE_EPARGNE;			
NUM_CE NUM	E_CLIENT	SOLDE	
1	8888	1000	
0			
1111	1111	100 8000	
1234	1234	8700	
SQL> select * from COMPTE_CHEQUE;			
NUM_CC NUM	_CLIENT	SOLDE	
,	0000	7000	
1 2	8888 1111		
	1234		
9999	9000	5000	
SQL>			

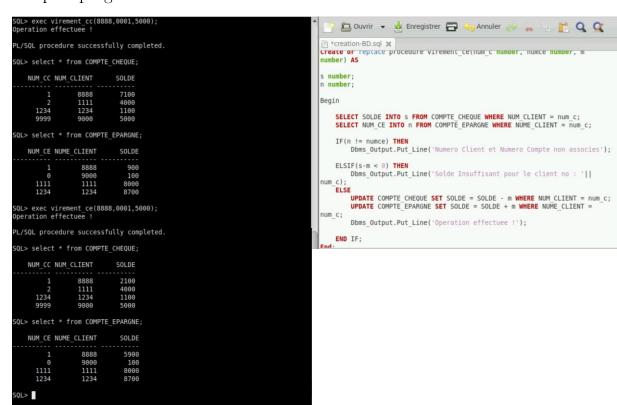
Erreur Numéro Client/Compte:

SQL> select * :	from COMPTE	E_EPARGNE;	
NUM_CE NUM	E_CLIENT	SOLDE	
1	8888	1000	
0	9000	100	
	1111		
	1234		
		73 710	
SQL> select *	from COMPTE	_CHEQUE;	
NUM_CC NUM	CLIENT	SOLDE	
1	8888	7000	
2	1111	4000	
1234	1234	1100	
9999	9000	5000	
PL/SQL procedure successfully completed. SQL> select * from COMPTE_EPARGNE;			
NUM_CE NUM	E_CLIENT	SOLDE	
1	8888	1000	
Θ	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		
9999	9000	5000	
SQL>			

(c)

Solde suffisant:

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



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
                 8888
                            2100
                 1111
                            4000
      1234
                 1234
                            1100
                 9000
SQL> select * from COMPTE EPARGNE;
   NUM CE NUME CLIENT
                             SOLDE
                  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
                  8888
                             5900
        0
                  9000
                              100
     1111
                  1111
                             8000
                             8700
     1234
                  1234
SQL> select * from COMPTE CHEQUE;
    NUM CC NUM CLIENT
                           SOLDE
                            2100
                            4000
      1234
                 1234
                             1100
      9999
                 9000
                            5000
SQL>
```

(d)

Crédit accepter:

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 0K
3 8888 0K
4 8888 K0

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.

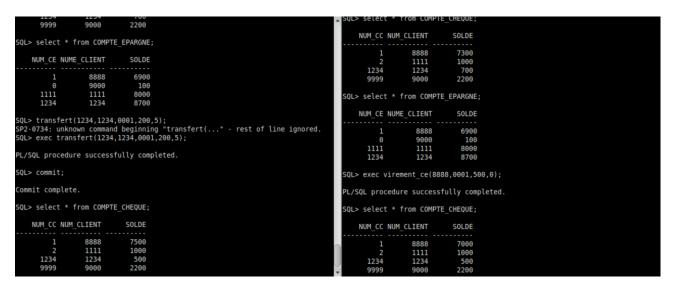
Session 1 Session 2

```
QL> exec transfert(1111,0002,0001,1000,0)
   NUM_CC_NUM_CLIENT
                                                                                     L/SQL procedure successfully completed.
                            SOLDE
                             4100
                                                                                    SQL> commit;
                 8888
                 1111
                             3000
                                                                                     Commit complete.
      1234
                             1100
                 1234
                             4000
      9999
                                                                                    SQL> exec transfert(1111,0002,0001,1000,4);
SQL> exec transfert(9000,9999,0001,1000,5);
                                                                                     PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
SQL> commit;
Commit complete.
SQL> select * from COMPTE_CHEQUE;
                                                                                     SQL> select * from COMPTE_CHEQUE;
   NUM CC NUM CLIENT
                                                                                        NUM CC NUM CLIENT
                                                                                                                 SOLDE
                            SOLDE
                             5100
                                                                                                                  6100
                 8888
                                                                                                      8888
                 1111
                             3000
                                                                                                                  2000
      1234
                                                                                           1234
                                                                                                      1234
                                                                                                                  1100
                 1234
                             1100
      9999
```

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

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

Session 1 Session 2



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 Session 2

```
QL> select * from COMPTE_EPARGNE;
   NUM CE NUME CLIENT
                                                                                            QL> exec traitement credit(9000,1000,0);
                                                                                            PL/SQL procedure successfully completed.
                                                                                            QL> select * from TRAITEMENT_CREDIT;
elect * from TRAITEMENT_CREDIT
SQL> exec transfert(9000,9999,0001,2000,5);
PL/SQL procedure successfully completed.
                                                                                            RROR at line 1:
RA-04044: procedure, function, package, or type is not allowed here
GQL> select * from COMPTE_CHEQUE;
   NUM CC NUM CLIENT
                              SOLDE
                                                                                            QL> select * from DECISION_CREDIT;
                  1111
1234
9000
                                                                                            UM DOSSIER NUM CLIENT DE
    select * from COMPTE EPARGNE
   NUM CE NUME CLIENT
                               SOLDE
```

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 3ème 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

 $O\dot{u}: \mathbf{RX} = \text{Row Exclusive}$

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_cp t;

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 NUME_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 NUME_CLIENT = num_c;
    LOCK TABLE COMPTE_EPARGNE IN EXCLUSIVE MODE;
    Dbms_Output.Put_Line('Operation effectuee !');
END IF;

End;</pre>
```

virement ce:

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

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;</pre>
```