M1 MIAGE FC/FA – 2015/2016

Système de Gestion de Bases de Données

septembre 2015

TP – transactions

Vous allez travailler par deux connectés simultanément sur deux machines voisines[1]. Les manipulations qui suivent utiliseront une table T ayant 2 colonnes numériques A et B.

Par la suite, j'utiliserai les abbréviations :

- `select T` pour `select * from T ;`
- `select T(A = 1)` pour `select * from T where A=1 ;`
- `insert (3,6)` pour `insert into T values (3,6);`
- `update(A <- A+1)` pour `update T set A = A+1 ;`
- `update(A <- 3)|(A = 4)` pour `update T set A=3 where A=4 ;`
- `delete(A = 4)` pour `delete from T where A=4 ;`

Pour commencer, créez la table T (une par binôme) et rendez-la accessible aux deux expérimentateurs (instruction `GRANT`) :

```
      session 1                 session 2
      ETUD_1                    ETUD_2
-------------------------------------------------------

create table T(
  a number,
  b number);

grant select, insert,
  delete, update
  on T
  to ETUD_2 ;
                                        create synonym T
                                        for ETUD_1.T
```

Insérez les lignes `(0,0)`, `(2,3)` et `(0,1)`, et faites un `commit` juste après l'insertion :

```
      session 1
------------------
insert (0,0)
insert (2,3)
insert (0,1)
commit
```

Transactions standards

Question 1 :

[1] vous pouvez aussi réaliser ce TP seul, en ouvrant deux connexions à SQLPlus

```
      session 1
------------------
insert (4,4)
insert ('a','b')
select T
```

Quelles instructions sont annulées ?

```
      session 1
------------------
insert (5,5)
rollback
select T
```

Et maintenant ?

Question 2 :

```
      session 1                  session 2
-----------------------------------------------------
   select T
                              update(A <- A+1)
   select T
                              commit
   select T
   commit
-----------------------------------------------------
```

Quel problème est mis en évidence ?

Question 3 :

```
      session 1                  session 2
-----------------------------------------------------
   update(A <- A+1)
   select T
                              insert(3,7)
                              select T
                              delete(A=2)
                              delete(A=1)
   commit
                              commit
-----------------------------------------------------
```

L'ordonnancement de ces deux transactions est-il sériable, c'est à dire équivalent à une exécution en série ?

Question 4 :

```
      session 1                session 2
-----------------------------------------------------
   update(A<-3)|(A=2)
                           update(B<-2)|(B=3)
                           select T
   select T
                           update(A<-3)|(A=2)
   update(B<-2)|(B=3)
   commit                  commit
-----------------------------------------------------
```

Quels sont les verrous posés par `session 1` et `session 2` avant le deuxième `update` de `session 2` ? Comment se terminent ces transactions ?

Question 5 : Avec Oracle, il existe une clause `for update` à l'instruction select. L'expérience suivante va vous permettre de comprendre son utilité (on reviendra sur cette clause quand on abordera de nouvelles spécificités de PL/SQL).

```
      session 1                   session 2
-----------------------------------------------------
   select T(B=7) for update
                               update(B<-10)|(B=0)
                               select T ;
                               update(A<-5)|(A=3)
   update(B<-6)|(B=7) ;
   commit

                               commit
                               select T
-----------------------------------------------------
```

Quel type de verrouillage est effectué ?

Le mode Read-Only

Question 6 :

```
      session 1                  session 2
-----------------------------------------------------
   set transaction read only;
   select T
                               select T
                               update(A<-A+1)
                               commit
   select T
                               select T
                               update(A<-A+1)
                               commit
   select T
   commit
   select T
-----------------------------------------------------
```

Quel problème est résolu ? Comment est-il résolu ?

Transaction serialisable

Question 7 :

```
      session 1                  session 2
-------------------------------------------------------
   select T
   update(A<-5 | A=6)
                              set transaction isolation level serializable ;
                              select T
                              update(B<-9 | B=10)
   select T
                             select T
   commit
                              select T
                              update(B<-3 | B=2)
                              select T ;
   select T
                              rollback ;
                              select T
-------------------------------------------------------
```

Que se passe-t-il à chaque étape de cet ordonnancement ?

Verrouillages explicites

Question 8 :

```
      session 1                          session 2
-------------------------------------------------------
   lock table T in exclusive mode ;
   select T
                                      select T
                                      select T for update
  update(A<-A+1)
  commit
                                      commit ;
-------------------------------------------------------
```

Comment expliquez-vous ce résultat ?

Question 9 :

```
      session 1                          session 2
-------------------------------------------------------
   lock table T in exclusive mode ;
   select T
                                      select T
                                      lock table T in exclusive mode nowait
  commit
                                      commit ;
-------------------------------------------------------
```

Comment expliquez-vous ce résultat ?

Les instructions du DDL

Question 10 :

```
      session 1                                  session 2
      ETUD_1                                     ETUD_2
-----------------------------------------------------------
   create synonym S for T
  grant select on S to ETUD_2
                                        select ETUD_1.S
   commit
   drop synonym S
                                        select ETUD_1.S
   rollback
   select S
-----------------------------------------------------------
```

Comment se comportent les instructions du DDL ?

Transactions en PL/SQL

Activez l'affichage dans SQLPLus grâce à la commande `SET SERVEROUTPUT ON`.

Supprimez toutes les données de `T` et ajoutez une contrainte de clé primaire :

```
alter table T add constraint T_Pkey primary key(a);
```

Pour toutes les questions suivantes, vous regarderez le contenu de la table `T` *après exécution du code PL/SQL et vous en déduirez ce qui s'est produit.*

Question 11 :

Exécutez le bloc anonyme suivant :

```
begin
  insert into T values (-1,-1) ;
  savepoint p1 ;
  insert into T values (-2,-2) ;
  rollback to p1 ;
  commit ;
end ;
```

Question 12 :

Créez la procédure `p` suivante :

```
create or replace procedure p(i1 NUMBER, i2 NUMBER) is
begin
  insert into T values (i1,i2);
  insert into T values (i1,i2+1);
  insert into T values (i1+1,i2+1);
  commit ;
end ;
```

Exécutez cette procédure par la commande `execute p(50,50)`.

Question 13 :

Modifiez la procédure `p` en ajoutant un traitement d'exception :

```
create or replace procedure p(i1 NUMBER, i2 NUMBER) is
begin
  insert into T values (i1,i2);
  insert into T values (i1,i2+1);
  insert into T values (i1+1,i2+1);
  commit ;
exception when others then dbms_output.put_line('pb insertion'); commit;
end ;
```

Exécutez à nouveau cette procédure par la commande `execute p(50,50)`.

Question 14 : Définissez les procédures `p` et `pp` suivantes :

```
create or replace procedure pp(i1 NUMBER, i2 NUMBER) is
begin
  insert into T values (i1,i2);
exception when others then dbms_output.put_line('pb insertion'); commit;
end ;

create or replace procedure p(i1 NUMBER, i2 NUMBER) is
begin
  insert into T values (i1,i2);
  pp(i1,i2);
  insert into T values (i1+1,i2+2);
  commit ;
exception when others then dbms_output.put_line('pb insertion'); commit;
end ;
```

Exécutez `p(60,60)`

Question 15 :

```
create or replace procedure pp(i1 NUMBER, i2 NUMBER) is
begin
  insert into T values (i1,i2);
  insert into T values (i1+1,i2+1);
exception when others then dbms_output.put_line('pb insertion pp');
end ;

create or replace procedure p(i1 NUMBER, i2 NUMBER) is
begin
  insert into T values (i1,i2);
  pp(i1+1,i2);
  insert into T values (i1+1,i2+2);
  commit ;
exception when others then dbms_output.put_line('pb insertion p');
                                rollback;
end ;
```

Exécutez `p(70,70)`.

Dans les questions suivantes, p reste inchangée.

Question 16 : Remplacez pp par cette procédure (on ajoute un commit) :

```
create or replace procedure pp(i1 NUMBER, i2 NUMBER) is
begin
  insert into T values (i1,i2);
  insert into T values (i1+1,i2+1);
  commit ;
exception
  when others then dbms_output.put_line('pb insertion pp');
end ;
```

Exécutez p(80,80).

Question 17 : Remplacez pp par cette procédure qui utilise une transaction autonome :

```
create or replace procedure pp(i1 NUMBER, i2 NUMBER) is
  pragma autonomous_transaction ;
begin
  insert into T values (i1,i2);
  insert into T values (i1+1,i2+1);
  commit ;
exception
  when others then dbms_output.put_line('pb insertion pp');
end ;
```

Exécutez p(90,90);

Retour sur le deadlock

L'utilisateur ETUD_1 créé une procédure lock_ligne_t :

```
create or replace
procedure lock_ligne_t(xa number, xb number) as
  deadlock exception ;
  pragma exception_init(deadlock, -60);
begin
  update t set a=a+1, b=b+1 where a=xa and b=xb ;
  dbms_output.put_line('ok insertion ligne');
  exception
  when deadlock then
    dbms_output.put_line('detection deadlock') ;
  when others then
    dbms_output.put_line('autre exception');
end lock_ligne_t;
```

Puis il donne les droits d'exécution à son binôme par un grant.

Question 18 :

```
      session 1                                session 2
      ETUD_1                                   ETUD_2
-----------------------------------------------------------
  lock_ligne_t(82,81)
                                       ETUD_1.lock_ligne_t(92,91)
                                       ETUD_1.lock_ligne_t(82,81)
   lock_ligne_t(92,91)
                                       commit
   commit
-----------------------------------------------------------
```

Est-ce que le traitement d'un deadlock est équivalent à un rollback ?