TP 1- Transactions sous Oracle **M2 IAFA**

Sous Oracle, une instruction DDL (`CREATE`, `ALTER`, `DROP`) ou DCL (`GRANT`, `REVOKE`, `COMMIT`, `ROLLBACK`) commence par valider la transaction précédente (`COMMIT`) puis s'exécute dans une nouvelle transaction qui se termine immédiatement par un `COMMIT` si l'instuction s'est bien passée, sinon la transaction doit être terminée manuellement. Tout autre suite d'instructions (terminée par un `COMMIT` ou un `ROLLBACK`) constitue une transaction.

Conformément à la norme SQL2, on peut définir le mode (*lecture seule* ou *lecture écriture*) d'une transaction par l'instruction :

```
SET TRANSACTION { READ ONLY | READ WRITE };
```

Il est également possible de spécifier le niveau d'isolation d'une transaction :

```
SET TRANSACTION ISOLATION LEVEL { SERIALIZABLE | READ COMMITTED };
```

Par défaut, le mode d'une transaction est `READ WRITE` et son niveau d'isolation est `READ COMMITTED`.
Dans la suite, nous allons déterminer expérimentalement la signification des différents modes et niveaux d'isolation et voir que leurs valeurs par défaut ne garantissent pas la sérialisabilité des transactions.

En d'autres termes :
Sous oracle, on peut définir le mode d'une transaction :

- Set transaction {read only | read write}
  - Read only : La transaction ne voit que les changements commis avant son début et ne peut pas effectuer de mise à jour. Elle ne voit aucune des modifications validées par d'autres transactions.
  - Read write : la transaction voit tous les changements validés et peut effectuer des mises à jour. Ceci est la valeur par défaut.

Il est possible de définir le niveau d'isolation d'une transaction :

- Set transaction isolation level {serializable | read commited}
  - Read commited : chaque instruction de la transaction voit ce qui était validé au moment où l'instruction a commencé à s'exécuter. Si cette instruction tente de verrouiller un tuple déjà verrouillé par une autre transaction, elle est bloquée jusqu'au déverrouillage ; quand elle se débloque elle réévalue complètement la sélection des tuples. Ceci est la valeur par défaut.
  - Serializable : Une transaction sérialisable voit la base telle qu'elle était validée quand elle a commencé. Elle ne voit aucune des modifications validées par d'autres transactions après son démarrage. Bien entendu elle voit ses propres modifications.

Attention, une instruction CREATE, ALTER, DROP, GRANT et REVOKE commence par valider la transaction précédente puis s'exécute dans une nouvelle. Si la transaction s'est bien passée, elle est validée sinon elle doit être terminée (validée ou non) manuellement.

2 Expériences

Il est préférable de travailler par deux connectés (`X` et `Y`) simultanément sur deux machines voisines. Vous pouvez aussi travailler seul mais il faut alors ouvrir deux connexions. Les manipulations qui suivent utiliseront une table `T` ayant 2 colonnes numériques `A` et `B`. On utilisera les abbréviations suivantes :

- `select T` pour `select * from T;`
- `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.`

Commencez par créer la table `T` (une par binôme) et rendez-la accessible aux deux utilisateurs :

```
session 1                                    |  session 2
---------------------------------------------------------------------------
create table T(a number, b number);          |
grant select, insert, delete, update on T to Y; |
                                             | create synonym T for X.T
---------------------------------------------------------------------------
```

Ensuite, insérez les lignes `(0,0)`, `(2,3)` et `(0,1)` et faites un `commit`.

2.1 Transactions standards

Expérience 1

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

Question 2.1 : Quel problème est mis en évidence ?

Expérience 2

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

Question 2.2 : L'ordonnancement de ces deux transactions est-il sérialisable ? Justifiez.

Expérience 3

```
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
---------------------------------------------------------------------------
```

Question 2.3 : Quels sont les verrous posés (types et objets) par session 1 et session 2 ? Comment se terminent ces transactions ? Pourquoi ?

Expérience 4

Avec Oracle, il existe une clause `for update` à l'instruction `select`. L'expérience suivante va vous permettre de comprendre son utilité.

```
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
---------------------------------------------------------------------------
```

Question 2.4 : Quel type de verrouillage est effectué ?

Expérience 5

```
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                                           |
---------------------------------------------------------------------------
```

Question 2.5 : Quel problème est résolu ? Comment est-il résolu ?

Expérience 6

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

Question 2.6 : Expliquez les résultats obtenus.