

Alexis
PLESSIER
27/12/2020

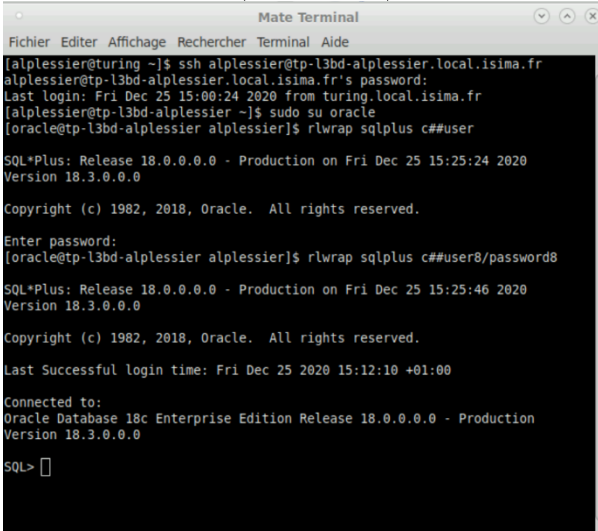
N3 Informatique

Implémentation des SGBD Rendu TP1

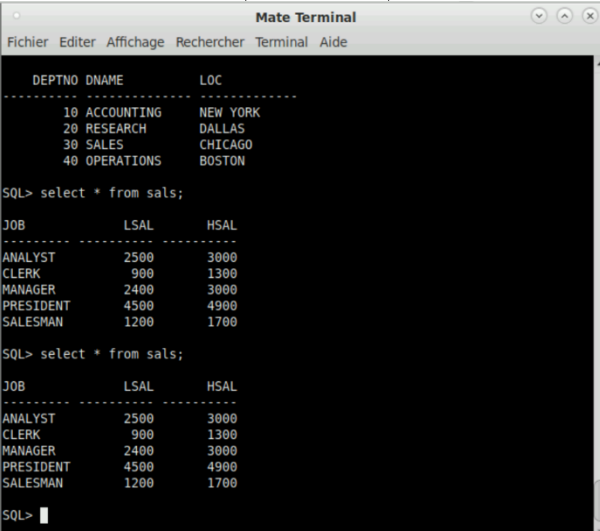
Exercice 1 :

1) Ouverture des deux sessions avec mon login « alplessier » et mon mot de passe.

(session 1)



(session 2)



2) Exécution des requêtes SQL à partir de la session 1, création des lignes dans la table EMP.

```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> INSERT INTO EMP VALUES (7000,'Petit Lion', 'SALESMAN',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);

1 row created.

SQL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2-1980','dd-mm-yyyy'),1600,300,20);
ERROR:
ORA-01756: quoted string not properly terminated

SQL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,20);

1 row created.

SQL>
```

(session1)

Mate Terminal

Fichier Editor Affichage Rechercher Terminal Aide

SOL>
SOL>
SOL>
SOL>
SOL>
SOL>
SOL>
SOL>
SOL>
SOL>
SOL> INSERT INTO EMP VALUES (7000,'Petit Lion', 'SALESMAN',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);

1 row created.

SOL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2-19801','dd-mm-yyyy'),1600,300,20);
ERROR:
ORA-01756: quoted string not properly terminated

SOL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,20);

1 row created.

(session 2)

Mate Terminal

Fichier Editor Affichage Rechercher Terminal Aide

7839 KING PRESIDENT 17-NOV-81 5000
10

EMPNO ENAME JOB MGR HIREDATE SAL COMM

DEPTNO

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0
30

7876 ADAMS CLERK 7788 23-MAY-87 1100
20

7900 JAMES CLERK 7698 03-DEC-81 950
30

EMPNO ENAME JOB MGR HIREDATE SAL COMM

DEPTNO

7902 FORD ANALYST 7566 03-DEC-81 3000
20

7934 MILLER CLERK 7782 23-JAN-82 1300
10

14 rows selected.

(session 1)

Mate Terminal

Fichier Editor Affichage Rechercher Terminal Aide

SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> INSERT INTO EMP VALUES (7000,'Petit Lion', 'SALESMAN',7902,to_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);

1 row created.

SQL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2-19801','dd-mm-yyyy'),1600,300,20);
ERROR:
ORA-01756: quoted string not properly terminated

SQL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,20);

1 row created.

SQL> commit;

Commit complete.

SQL> █

(session 2)

Mate Terminal

Fichier Editor Affichage Rechercher Terminal Aide

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM
7900	JAMES	CLERK	7698	03-DEC-81	950	
30						

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM

DEPTNO	-----					
7902	FORD	ANALYST	7566	03-DEC-81	3000	
20						
7934	MILLER	CLERK	7782	23-JAN-82	1300	
10						
7000	Petit Lion	SALESMAN	7902	17-DEC-80	800	
20						

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM

DEPTNO	-----					
7001	Chaussette	SALESMAN	7698	20-FEB-81	1600	300
20						

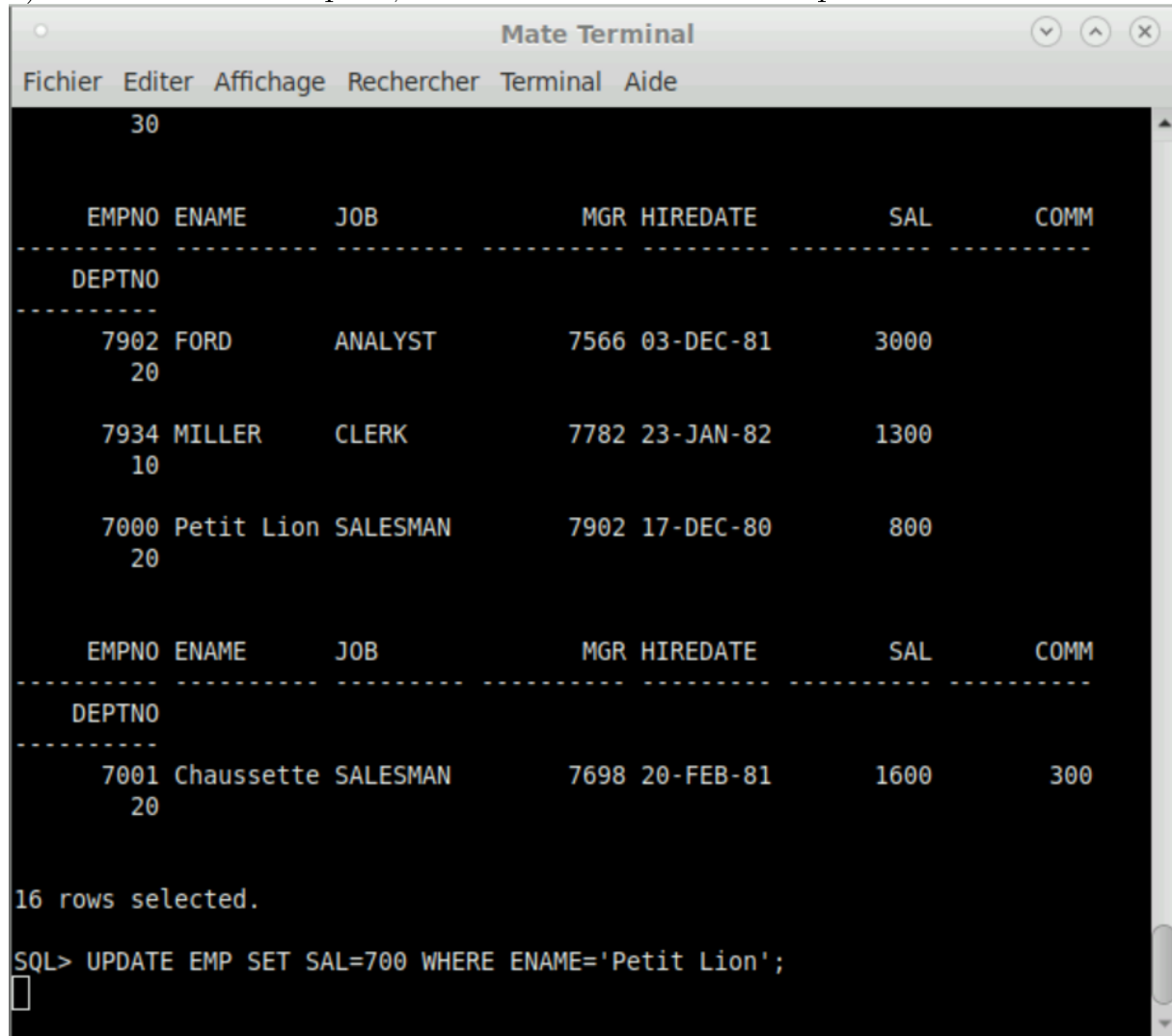
16 rows selected.

SQL> █

5)

```
SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';  
1 row updated.
```

6) En faisant cette requête, le SGBD se met en état bloquant dans la session 2.



The screenshot shows a terminal window titled "Mate Terminal" with a menu bar containing "Fichier", "Editer", "Affichage", "Rechercher", "Terminal", and "Aide". The terminal displays the following SQL queries and their results:

```
30  
  
      EMPNO ENAME      JOB      MGR HIREDATE      SAL      COMM  
-----  
DEPTNO  
-----  
      7902 FORD      ANALYST      7566 03-DEC-81      3000  
      20  
  
      7934 MILLER      CLERK      7782 23-JAN-82      1300  
      10  
  
      7000 Petit Lion SALESMAN      7902 17-DEC-80      800  
      20  
  
      EMPNO ENAME      JOB      MGR HIREDATE      SAL      COMM  
-----  
DEPTNO  
-----  
      7001 Chaussette SALESMAN      7698 20-FEB-81      1600      300  
      20  
  
16 rows selected.  
  
SQL> UPDATE EMP SET SAL=700 WHERE ENAME='Petit Lion';  
█
```

7) Après le COMMIT dans la session 1, la session 2 arrive à faire le UPDATE.

(session 1)

```
Mate Terminal
Fichier Editer Affichage Rechercher Terminal Aide

SQL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2
-1980','dd-mm-yyyy'),1600,300,20);
ERROR:
ORA-01756: quoted string not properly terminated

SQL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2
-1981','dd-mm-yyyy'),1600,300,20);
1 row created.

SQL> commit;
Commit complete.

SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';
0 rows updated.

SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';
1 row updated.

SQL> commit;
Commit complete.

SQL>
```

(session 2)

```
Mate Terminal
Fichier Editer Affichage Rechercher Terminal Aide

EMPNO ENAME      JOB      MGR HIREDATE      SAL      COMM
-----
7902 FORD        ANALYST   7566 03-DEC-81    3000
7934 MILLER      CLERK     7782 23-JAN-82    1300
7000 Petit Lion SALESMAN  7902 17-DEC-80    800
7001 Chaussette SALESMAN  7698 20-FEB-81    1600      300

16 rows selected.

SQL> UPDATE EMP SET SAL=700 WHERE ENAME='Petit Lion';
1 row updated.

SQL>
```

Or quand on fait un SELECT du salaire de 'Petit Lion' dans la session 2, on le salaire à bien été changer mais quand on fait la même requête dans la session 1, l'UPDATE de la session 2 n'est pas encore effectif dans la session 1.

(session 1)

```
Mate Terminal
Fichier Editer Affichage Rechercher Terminal Aide

SQL> INSERT INTO EMP VALUES (7001,'Chaussette', 'SALESMAN',7698,to_date('20-2
-1981','dd-mm-yyyy'),1600,300,20);
1 row created.

SQL> commit;
Commit complete.

SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';
0 rows updated.

SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';
1 row updated.

SQL> commit;
Commit complete.

SQL> select SAL from EMP WHERE ENAME='Petit Lion';

      SAL
-----
      1000

SQL>
```

(session 2)

```
Mate Terminal
Fichier Editer Affichage Rechercher Terminal Aide

7934 MILLER      CLERK     7782 23-JAN-82    1300
7000 Petit Lion SALESMAN  7902 17-DEC-80    800
7001 Chaussette SALESMAN  7698 20-FEB-81    1600      300

16 rows selected.

SQL> UPDATE EMP SET SAL=700 WHERE ENAME='Petit Lion';
1 row updated.

SQL> select SAL from EMP WHERE ENAME='Petit Lion';

      SAL
-----
       700

SQL>
```

8) Après le COMMIT dans la session 2, l'UPDATE de la session 2 écrase l'UPDATE présente dans la session 1 et met bien à jour sa valeur.

(session 1)

```

SQL> commit;
Commit complete.
SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';
0 rows updated.
SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';
1 row updated.
SQL> commit;
Commit complete.
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      1000
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      700
SQL>

```

(session 2)

```

DEPTNO
-----
7001 Chaussette SALESMAN    7698 20-FEB-81    1600    300
20
16 rows selected.
SQL> UPDATE EMP SET SAL=700 WHERE ENAME='Petit Lion';
1 row updated.
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      700
SQL> commit;
Commit complete.
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      700
SQL>

```

9) Le SELECT FOR UPDATE va se réserver l'accès aux lignes que l'on a sélectionné à partir de la session 1, c'est-à-dire que personne ne pourra modifier les valeurs tant qu'il n'y a pas de COMMIT ou ABORT. Donc avec la session 2, quand on va vouloir faire un UPDATE sur la même ligne ciblée par la session 1 (avec le SELECT FOR UPDATE), la requête SQL va se mettre en état bloquant (ligne 'Petit Lion' pour l'exemple).

(session 1)

```

0 rows updated.
SQL> UPDATE EMP SET SAL = SAL+200 WHERE ENAME='Petit Lion';
1 row updated.
SQL> commit;
Commit complete.
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      1000
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      700
SQL> select SAL from EMP WHERE ENAME='Petit Lion' FOR UPDATE;
      SAL
-----
      700
SQL>

```

(session 2)

```

DEPTNO
-----
7001 Chaussette SALESMAN    7698 20-FEB-81    1600    300
20
16 rows selected.
SQL> UPDATE EMP SET SAL=700 WHERE ENAME='Petit Lion';
1 row updated.
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      700
SQL> commit;
Commit complete.
SQL> select SAL from EMP WHERE ENAME='Petit Lion';
      SAL
-----
      700
SQL> UPDATE EMP SET SAL=888 WHERE ENAME='Petit Lion';

```

Quand on fait un COMMIT à partir de la session 1, l'UPDATE que l'on a tenté de faire à partir de la session 2 va prendre effet sur la session 2. Ensuite, après un COMMIT sur la session 2, l'UPDATE va prendre effet sur les deux sessions.

The image shows two side-by-side terminal windows titled "Mate Terminal". The left window contains the following SQL commands and their outputs:

```
SQL> select SAL from EMP WHERE ENAME='Petit Lion';

      SAL
-----
      1000

SQL> select SAL from EMP WHERE ENAME='Petit Lion';

      SAL
-----
      700

SQL> select SAL from EMP WHERE ENAME='Petit Lion' FOR UPDATE;

      SAL
-----
      700

SQL> commit;

Commit complete.

SQL> select SAL from EMP WHERE ENAME='Petit Lion';

      SAL
-----
      888

SQL>
```

The right window contains the following SQL commands and their outputs:

```
      700

SQL> commit;

Commit complete.

SQL> select SAL from EMP WHERE ENAME='Petit Lion';

      SAL
-----
      700

SQL> UPDATE EMP SET SAL=888 WHERE ENAME='Petit Lion';

1 row updated.

SQL> commit
2
SQL> commit;

Commit complete.

SQL> select SAL from EMP WHERE ENAME='Petit Lion';

      SAL
-----
      888

SQL>
```

Exercice 2 :

Scénario 1 :

(session 1)

```
SQL> SELECT SAL FROM EMP WHERE EMPNO = 7369;

      SAL
-----
      800
```

(session 2)

```
SQL> UPDATE EMP SET SAL = 802 WHERE EMPNO = 7369;

1 row updated.

SQL> COMMIT;

Commit complete.
```

(session 1)

```
SQL> SELECT SAL FROM EMP WHERE EMPNO = 7369;

      SAL
-----
      802

SQL>
```

Dans un premier temps, on demande le salaire de l'employé n°7369 qui est 800. Avec la session 2 on définit le salaire de ce même employé à 802 puis on COMMIT sur cette même session. En revenant à la session 1 et en affichant le salaire de cet employé, nous nous rendons compte que son salaire est de 802, donc que la modification de la session 2 est bel et bien visible. Ceci grâce au COMMIT effectué sur la session 2 avant le SELECT de la session 1.

Scénario 2 :

(session 1)

```
SQL> UPDATE EMP SET SAL = 801 WHERE EMPNO = 7369;  
1 row updated.
```

(session 2)

```
SQL> SELECT SAL FROM EMP WHERE EMPNO = 7369;  
  
      SAL  
-----  
      802
```

(session 1)

```
SQL> COMMIT;  
  
Commit complete.
```

(session 2)

```
SQL> SELECT SAL FROM EMP WHERE EMPNO = 7369;  
  
      SAL  
-----  
      801  
  
SQL> █
```

On fait un UPDATE sur le salaire de l'employé n°7369 avec la session 1, puis on regarde le salaire de ce même employé en session 2, et on remarque que la mise à jour de la valeur n'est pas effective (en session 2). On fait par la suite un commit en session 1 et réaffichons le salaire de l'employé en session 2 et là, la valeur est bien mise à jour. Ce phénomène est possible grâce au COMMIT effectué en session 1 après la modification de valeur du salaire.

Scénario 3 :

(session 1)

```
SQL> UPDATE EMP SET SAL = 803 WHERE EMPNO = 7369;  
  
1 row updated.  
  
SQL> █
```

(session 2)

->

(session 1)

```
SQL> UPDATE EMP SET SAL = 804 WHERE EMPNO = 7369;  
  
1 row updated.  
  
SQL> █
```

```
SQL> COMMIT;  
  
Commit complete.  
  
SQL> █
```

(session 2)

```
SQL> COMMIT;  
  
Commit complete.  
  
SQL> █
```

La session 1 met à jour le salaire de l'employé n°7369 à 803.

La session 2 met à jour le salaire de l'employé n°7369 à 804. La ligne est en attente car cette valeur est en cours de modification sur une autre session.

Puis on fait un COMMIT en session 1 puis un COMMIT en session 2. Les deux sessions vont donc modifier le salaire d'origine qui est de 801. La session 1 est la première à COMMIT, le salaire sera alors de 803 puis viens le COMMIT de la session 2 et modifie la valeur qui passe de 803 en 804.

Scénario 4 :

(session 1)

```
SQL> UPDATE EMP SET SAL = 805 WHERE EMPNO = 7369;  
  
1 row updated.  
  
SQL>
```

(session 2)

```
SQL> UPDATE EMP SET SAL = 1300 WHERE EMPNO = 7521;  
  
1 row updated.  
  
SQL>
```

(session 1)

```
SQL> UPDATE EMP SET SAL = 1300 WHERE EMPNO = 7521;  
█
```

(session 2)

(attente)

SQL> UPDATE EMP SET SAL = 1300 WHERE EMPNO = 7521; 1 row updated. SQL> UPDATE EMP SET SAL = 805 WHERE EMPNO = 7369; █	UPDATE EMP SET SAL = 1300 WHERE EMPNO = 7521 * ERROR at line 1: ORA-00060: deadlock detected while waiting for resource SQL>
--	--

(session 1)

SQL> UPDATE EMP SET SAL = 805 WHERE EMPNO = 7369; 1 row updated. SQL> █	SQL> ROLLBACK; Rollback complete. SQL> █
---	--

Ici, les deux sessions vont se bloquées car elles essaient de modifier les mêmes valeurs. La session 1 veut modifier le salaire de l'employé n°7369 et la session 2 le salaire de l'employé n°7521. Jusqu'ici chaque session attende un COMMIT pour bien terminer leurs transactions. Cependant, la session 1 va essayer à son tour de modifier le salaire de l'employé n°7521 et la session 2 le salaire de l'employé n°7369, les deux sessions se bloquent et l'erreur DEADLOCK est renvoyée. La session 1 effectue un ROLLBACK et

reviens aux dernières valeurs enregistrées dans le journal avant la mise à jour. La session 2 est donc débloquée et peut effectuer ses mises à jour.

Exercice 3 :

- Empêcher les lectures non reproductibles :

- Si l'utilisateur ne modifie aucune donnée :

Il suffit de la mettre en mode "read only" avec la commande "SET transaction READ ONLY".

- Si l'utilisateur modifie des données :

On passe en mode serializable avec la commande "SET transaction isolation level SERIALIZABLE".

- Empêcher les lignes fantômes :

- Si l'utilisateur ne modifie aucune donnée :

On peut mettre la transaction en read only aussi.

- Si l'utilisateur modifie des données :

On peut bloquer les tables ou se mettre en mode serializable.

Read Only :

1)

(session 1)

```
SQL> SET transaction read only;  
Transaction set.  
SQL> █
```

2)

(session 2)

```
SQL> update emp set sal= 1000 where empno = 7369;  
1 row updated.  
SQL>
```

3)

(session 2)

```
SQL> commit;  
Commit complete.  
SQL> █
```

Non on ne voit pas les modifications depuis la 1^{ère} session :

```
SQL> select sal from emp where empno=7369;  
  
      SAL  
-----  
      804  
  
SQL> █
```

4) Ce n'est pas possible car nous sommes en Read Only dans la session 1 (depuis la question 1).

```
SQL> update emp set sal= 888 where empno=7369;  
update emp set sal= 888 where empno=7369  
      *  
ERROR at line 1:  
ORA-01456: may not perform insert/delete/update operation inside a READ ONLY  
transaction  
  
SQL>
```

5) Si le mode “Read Only” n’avait pas été activé dans la première session, on aurait pu voir les données modifiées après le COMMIT fait sur la deuxième session.

```
SQL> commit;

Commit complete.

SQL> select sal from emp where empno=7369;

      SAL
-----
     1000

SQL> _
```

On voit ici qu'après le COMMIT sur la session 1 (*aussi transaction 1*), on peut voir les modifications faites en session 2 précédemment.

Mode sérialisable

1)2)

(T1)

```
SQL> set transaction isolation level serializable;

Transaction set.

SQL>
```

3)

(T1)

```
SQL> SELECT ENAME, SAL FROM EMP;

ENAME          SAL
-----
SMITH          1000
ALLEN          1600
WARD           1300
JONES          2975
MARTIN         1250
BLAKE          2850
CLARK          2450
SCOTT          3000
KING           5000
TURNER         1500
ADAMS          1100

ENAME          SAL
-----
JAMES           950
FORD            3000
MILLER          1300
Petit Lion       500
Chaussette      1600

16 rows selected.

SQL> █
```

4)

(T2)

```
SQL> update emp set sal=888 where ename='WARD';  
  
1 row updated.  
  
SQL>
```

5) On n'a pas encore COMMIT l'update fait dans T2 donc c'est logique que l'on ne voit pas la modification faite sur l'employé "WARD".

(T1)

```
SQL> select ename,sal from emp
```

ENAME	SAL
SMITH	1000
ALLEN	1600
WARD	1300

6)

(T2)

```
SQL> commit;  
  
Commit complete.  
  
SQL> █
```

7) On ne voit toujours pas la modification à partir de T1 même après le COMMIT fait en T2, car on a isolé en mode sérialisable T1 au tout début.

(T1)

```
SQL> select ename,sal from emp
```

ENAME	SAL
SMITH	1000
ALLEN	1600
WARD	1300

8) On ne peut pas modifier la valeur du salaire de l'employé car on a mis le mode sérialisable sur T1 dès le début.

(T1)

```
SQL> update emp set sal=889 where ename='WARD';
update emp set sal=889 where ename='WARD'
      *
ERROR at line 1:
ORA-08177: can't serialize access for this transaction
```

9) Même problème, si l'on veut modifier des valeurs sur la table EMP, on doit faire un COMMIT sur T1. Cependant la modification de valeurs entrées sur cette section oracle ne pose aucun problème (cf. Petit Lion et Chaussette).

(T1)

```
SQL> update emp set sal=889 where ename='Petit Lion';
1 row updated.

SQL> update emp set sal=889 where ename='JAMES';
update emp set sal=889 where ename='JAMES'
      *
ERROR at line 1:
ORA-08177: can't serialize access for this transaction

SQL> update emp set sal=889 where ename='Chaussette';
1 row updated.

SQL> update emp set sal=889 where ename='MILLER';
update emp set sal=889 where ename='MILLER'
      *
ERROR at line 1:
ORA-08177: can't serialize access for this transaction

SQL> █
```

Avec le mode par défaut d'Oracle, T1 aurait vu la modification faite par T2 après le COMMIT. T1 aurait aussi pu modifier le salaire de cet employé.