Rapport TP03

BD1

NAAJI Dorian & ARMANET Nathan – 3A INFO GROUPE 2
POLYTECH LYON
DORIAN.NAAJI@ETU.UNIV-LYON1.FR
NATHAN.ARMANET@ETU.UNIV-LYON1.FR

1. Partie 1 : Transactions

1.1. Atomicité d'une transaction courante

1.1.1. Question 1

« ROLLBACK » a supprimé tous les changements effectués à la table (insertion, modification et suppression de données) mais pas la table elle-même.

1.1.2. Question 2

« COMMIT » a enregistré les modifications apportées à la table, ce qui à empêcher « ROLLBACK » de les supprimer.

1.1.3. Question 3

« EXIT » ou « QUIT » ferme simplement la connexion à la base de données sans apporter de modification à celle-ci.

1.1.4. Question 4

La fermeture brutale de la session est comme un « ROLLBACK » ; elle supprimer toutes les modifications de données non enregistrées.

1.1.5. Question 5

Cette fois ci, le « ROLLBACK » ne supprime rien. Cela est dû au fait que l'on a modifier la table en elle-même (ajout, modification ou suppression de colonnes ou contraintes). Ces modifications enregistrent les données tel qu'elles sont avant de modifier la table.

1.1.6. Question 6

Une transaction courante est un changement dans les données d'une table. Elles sont validées par « COMMIT » et annulé par « ROLLBACK ». Toutes modification de la table entraine un « COMMIT ».

1.2. Plusieurs sessions sur un seul compte de BD et transactions concurrentes

1.2.1. Question 1

On constate que le contenu de la base est le même quel que soit la fenêtre utilisée

1.2.2. Question 2

On voit juste les transactions effectuées à partir de la même fenêtre

1.2.3. Question 3

On voit les modifications de la table initiale apporté par l'utilisateur qui a créé la nouvelle table.

On peut voir la nouvelle table sur les deux fenêtres.

On voit seulement les transactions effectuer à partir de notre fenêtre

1.2.4. Question 4

La suppression de la table fonctionne.

1.2.5. Question 5

Seulement le premier utilisateur essayant d'insérer les données y parvient. Le deuxième utilisateur effectuant une insertion rencontre en effet un blocage « infini ». Une fois que le premier utilisateur effectue un ROLLBACK, l'insertion du deuxième utilisateur est instantanément effectué.

1.2.6. Question 6

On constate que la commande « EXIT » a commit les modifications.

1.2.7. Question 7

En sortant normalement, la dernière transaction a en effet été validée.

1.2.8. Question 8

On insère une ligne puis on crée une table ; un COMMIT est donc effectué et la ligne insérée dans la première table ne peut plus être ROLLBACK.

On insère ensuite une ligne dans la nouvelle table que l'on vient de créer, puis on ROLLBACK. La transaction ajoutant la ligne à la nouvelle table est ainsi annulée, mais la ligne ajoutée à la première table est toujours présente.

1.2.9. Question 9

La dernière ligne insérée est toujours présente. On en déduit que la commande « DROP TABLE » effectue un commit.

1.3. Droits/privilèges entre deux comptes d'une même base de données

1.3.1. Question 1

On se donne les privilèges et on peut vérifier qu'ils ont bien été données grâce aux requêtes

*« SELECT * FROM ALL_OBJECTS WHERE OWNER = 'INI3A06'; »*

*« SELECT * FROM ALL_TABLES WHERE OWNER = 'INI3A06'; »*

*« SELECT * FROM USER_TAB_PRIVS WHERE OWNER = 'INI3A06'; »*

*« select * from INI3A06.TRANSACTION »*

1.3.2. Question 2

Quand l'autre groupe fait une mise à jour sur sa table puis effectue un COMMIT, on peut voir les modifications.

1.3.3. Question 3

On ne parvient pas à insérer une mise à jour sur la table de l'autre groupe car nous n'avons pas le droit d'insérer de données dans cette dernière.

1.3.4. Question 4

Avec le droit d'insertion, notre commande fonctionne désormais.

1.3.5. Question 5

On peut réaliser la requête suivante pour la jointure :

*« SELECT * FROM INI3A06.TRANSACTION*

INNER JOIN TESTQ5_TP3 ON TESTQ5_TP3.id = INI3A06.TRANSACTION.ID_TRANSACTION; »

2. Partie 2 : PLSQL

2.1. Copier dans votre compte les trois table Dept, Emp et Salgrade de Scott.

On effectue les requêtes suivantes :

*« CREATE TABLE EMP AS SELECT * FROM SCOTT.emp; »*

*« CREATE TABLE DEPT AS SELECT * FROM SCOTT.Dept; »*

*« CREATE TABLE SalGrade AS SELECT * FROM SCOTT.SalGrade; »*

2.2. Donner les requêtes SQL pour les questions suivantes :

2.2.1. Donner le nom des employés dirigés directement par 'King'.

On effectue la requête :

« select E1.* from EMP E1

join EMP E2 on E2.DEPTNO = E1.DEPTNO

where E2.ENAME = 'KING'

and E1.ENAME != 'KING'; »

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7566	JONES	MANAGER	7839	02/04/81	2975	(null)	20
2	7698	BLAKE	MANAGER	7839	01/05/81	2850	(null)	30
3	7782	CLARK	MANAGER	7839	09/06/81	2450	(null)	10

FIGURE 1 : RESULTAT DE LA PREMIERE REQUETE

2.2.2. Donner le nom des employés qui dépendent (directement ou non) de Jones.

On effectue la requête :

« SELECT EMP.*

FROM EMP

WHERE EMP.ENAME <> 'JONES'

START WITH EMP.MGR = (SELECT EMP.EMPNO FROM EMP WHERE ENAME = 'JONES')

CONNECT BY EMP.MGR = prior EMP.EMPNO; »

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7788	SCOTT	ANALYST	7566	19/04/87	3000	(null)	20
2	7876	ADAMS	CLERK	7788	23/05/87	1100	(null)	20
3	7902	FORD	ANALYST	7566	03/12/81	3000	(null)	20
4	7369	SMITH	CLERK	7902	17/12/80	800	(null)	20

FIGURE 2 : RESULTAT DE LA SECONDE REQUETE

2.2.3. Donner le nom des employés dont dépend (directement ou non) Jones.

On effectue la requête :

« SELECT EMP.*

FROM EMP

WHERE EMP.ENAME <> 'JONES'

START WITH EMP.ENAME = 'JONES'

connect by EMP.EMPNO = prior EMP.MGR; »

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7788	SCOTT	ANALYST	7566	19/04/87	3000	(null)	20
2	7876	ADAMS	CLERK	7788	23/05/87	1100	(null)	20
3	7902	FORD	ANALYST	7566	03/12/81	3000	(null)	20
4	7369	SMITH	CLERK	7902	17/12/80	800	(null)	20

FIGURE 3 : RESULTAT DE LA TROISIEME REQUETE

2.2.4. Donner le nom des employés dépendant de Blake, sauf Blake lui-même.

On effectue la requête :

« SELECT EMP.*

FROM EMP

WHERE EMP.ENAME <> 'BLAKE'

START WITH EMP.MGR = (SELECT EMP.EMPNO FROM EMP WHERE EMP.ENAME = 'BLAKE')

CONNECT BY EMP.MGR = prior EMP.EMPNO; »

	E...	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
2	7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
3	7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
4	7844	TURNER	SALESMAN	7698	08/09/81	1500	0	30
5	7900	JAMES	CLERK	7698	03/12/81	950	(null)	30

FIGURE 4 : RESULTAT DE LA QUATRIEME REQUETE

2.2.5. Donner le nom des employés qui dépendent de King, sauf ceux qui dépendent de Blake.

On effectue la requête :

« SELECT EMP.*

FROM EMP

WHERE EMP.ENAME <> 'KING'

START WITH EMP.MGR = (SELECT EMP.EMPNO FROM EMP WHERE EMP.ENAME = 'KING')

CONNECT BY EMP.MGR = prior EMP.EMPNO

MINUS

```
SELECT EMP.*

FROM EMP

WHERE EMP.ENAME <> 'BLAKE'

START WITH EMP.MGR = (SELECT EMP.EMPNO FROM EMP WHERE EMP.ENAME = 'BLAKE')

CONNECT BY EMP.MGR = prior EMP.EMPNO; »
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	17/12/80	800	(null)	20
2	7566	JONES	MANAGER	7839	02/04/81	2975	(null)	20
3	7698	BLAKE	MANAGER	7839	01/05/81	2850	(null)	30
4	7782	CLARK	MANAGER	7839	09/06/81	2450	(null)	10
5	7788	SCOTT	ANALYST	7566	19/04/87	3000	(null)	20
6	7876	ADAMS	CLERK	7788	23/05/87	1100	(null)	20
7	7902	FORD	ANALYST	7566	03/12/81	3000	(null)	20
8	7934	MILLER	CLERK	7782	23/01/82	1300	(null)	10

FIGURE 5 : RESULTAT DE LA CINQUIEME REQUETE

2.3. Écrire une fonction PLSQL de paramètre un numéro de département et qui retourne le nombre d'employés de ce département. On pourra typer le numéro de département par *Emp.DeptNo%TYPE*.

```
-- déclaration fonction
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION getNbrEmp(p_noDept IN EMP.DeptNo%TYPE)
RETURN EMP.DeptNo%TYPE
IS
    l_empNumber EMP.DeptNo%TYPE;
BEGIN
    SELECT COUNT (*) INTO l_empNumber
    FROM EMP
    WHERE EMP.DEPTNO = p_noDept;
    RETURN l_empNumber;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Pas d''employés trouvés pour ce département.');
    RETURN null;
END;


-- test de la fonction
SET SERVEROUTPUT ON
BEGIN
    DBMS_OUTPUT.PUT_LINE(getNbrEmp(20));
END;
```

Cette fonction nous donne :

3 employés dans le département n°10.

5 employés dans le département n°20.

6 employés dans le département n°30.

0 employés dans le département n°40.

2.4. Ajouter la colonne NbEmps dans votre table Dept, qui contiendra le nombre d'employés de chaque département. On mettra à jour NbEmps de deux façons possibles :

2.4.1. En utilisant la fonction stockée définie dans la question précédente

```
ALTER TABLE Dept ADD NbEmps INT;
```

```
UPDATE DEPT
SET NBEMPS = getNbrEmp(10)
WHERE DEPTNO = 10;

UPDATE DEPT
SET NBEMPS = getNbrEmp(20)
WHERE DEPTNO = 20;

UPDATE DEPT
SET NBEMPS = getNbrEmp(30)
WHERE DEPTNO = 30;

UPDATE DEPT
SET NBEMPS = getNbrEmp(40)
WHERE DEPTNO = 40;
```

2.4.2. Sans l'utiliser mais en utilisant un curseur associé à Dept suivant l'exemple suivant :

```
DECLARE
CURSOR c_noDep IS SELECT DEPTNO FROM DEPT;
l_no DEPT.DeptNo%TYPE;
BEGIN
    FOR l_noDep IN c_noDep LOOP

--DBMS_OUTPUT.PUT_LINE(l_noDep.DEPTNO);
       UPDATE DEPT SET NBEMPS = (SELECT COUNT (*) FROM EMP WHERE EMP.DEPTNO = l_noDep.DEPTNO);
    END LOOP;
END;
```