SGBD : BASES DE DONNÉES AVANCÉES [M3106C]

TD $N^0$4 - ACCÈS CONCURRENTS SOUS POSTGRESQL

Objectifs

- Modes d'Isolation : READ COMMITED / SERIALIZABLE
- Transaction sérialisable
- Blocage mutuel

Corrigés

Exercice I :

Question 1.1.

2 transactions :

(1) BEGIN TRANSACTION ; .... COMMIT ;

(2) SELECT ... 'P4' ;

Question 1.2.

Une transaction est une unité atomique (exécution) qui préserve la consistence de la bd. Ici la deuxième insertion viole une contrainte d'intégrite donc la transaction est avortée (aborted).

```
-----------------------------------------------
# BEGIN TRANSACTION;
BEGIN
# INSERT INTO Commande
  VALUES ('F2','P4',25);
INSERT 221664 1
# INSERT INTO Commande
  VALUES ('F2','P8',25);
ERROR:  commande_pi_numero_fk referential
      integrity violation - key referenced
       from commande not found in piece
# COMMIT;
ROLLBACK
# SELECT * FROM commande
```

Date: 10 septembre 2014.

Hocine ABIR - IUT Villetaneuse .

```
    WHERE co_piece='P4';
 co_fournisseur | co_piece | co_quantite
----------------+----------+-------------
(0 rows)
```

Question 1.3.

```
# SELECT * FROM commande
   WHERE co_piece='P4';
NOTICE:  current transaction is aborted,
             queries ignored until
             end of transaction block
*ABORT STATE*
```

Question 1.4.

```
# begin transaction;
BEGIN
# INSERT INTO Commande
           VALUES ('F2','P4',25);
INSERT 1
# SELECT * FROM commande
     WHERE co_piece='P4';
 co_fournisseur | co_piece | co_quantite
----------------+----------+-------------
 F2             | P4       |          25
(1 row)

# INSERT INTO Commande
#          VALUES ('F2','P8',25);
ERROR:  commande_pi_numero_fk referential
   integrity violation - key referenced from
   commande not found in piece
# commit;
ROLLBACK
# SELECT * FROM commande
      WHERE co_piece='P4';
 co_fournisseur | co_piece | co_quantite
----------------+----------+-------------
(0 rows)
```

Question 1.5.

Il suffit de décrire un ordonnancement dans lequel une première transaction T_i lit (`SELECT`) deux fois les mêmes tuples et une deuxième transaction T_j qui met à jour (`UPDATE, DELETE, INSERT`) des tuples lus par T_i et se termine (commit) entre les deux lectures de T_i.

Exemple		
Etape	Terminal 1	Terminal 2
1	BEGIN TRANSACTION ;	
2	SELECT * FROM fournisseur ;	
3		UPDATE fournisseur SET fo_categorie=fo_categorie+2 ;
4	SELECT * FROM fournisseur ;	
5	COMMIT ;	

- READ COMMITTED : La première transaction T_i voit les mises à jour de la deuxième transaction T_j. Dans l'exemple ci-dessus, les lectures (2) et (4) seront différentes.
- SERIALIZABLE : La transaction T_i ne voit pas les mises à jour effectuées par la transaction T_j . Dans l'exemple ci-dessus, les lectures (2) et (4) seront identiques.

Exercice II :

Question 2.1.

6 transactions

Question 2.2.

- (1-4) la transaction Etape 1 et la transaction Etape 2 lisent les mêmes données car la requête Etape 3 fait partie d'une transaction non terminée : transaction Etape 2 ne voient pas les maj de la requête Etape 3.
- (4-6) la transaction Etape 6 voient les maj de la requête Etape 3 qui fait partie d'une transaction terminée en Etape 5
- (6-10) la transaction Etape 6 et la transaction Etape 10 lisent les mêmes données car la requête Etape 8 fait partie d'une transaction annulée.

Etape	O^1	
	Terminal 1	Terminal 2
1	SELECT fo_nom,fo_categorie FROM fournisseur WHERE fo_nom = 'Dupont'; fo_nom | fo_categorie ------+------- Dupont | 20 Dupont | 20 (2 rows)	
2		BEGIN TRANSACTION; BEGIN
3		UPDATE fournisseur SET fo_categorie=fo_categorie+2 WHERE fo_nom='Dupont'; UPDATE 2
4	SELECT fo_nom,fo_categorie FROM fournisseur WHERE fo_nom = 'Dupont'; fo_nom | fo_categorie ------+------- Dupont | 20 Dupont | 20 (2 rows)	
5		COMMIT; COMMIT
6	SELECT fo_nom,fo_categorie FROM fournisseur WHERE fo_nom = 'Dupont'; fo_nom | fo_categorie ------+------- Dupont | 22 Dupont | 22 (2 rows)	
7		BEGIN TRANSACTION;
8		UPDATE fournisseur SET fo_categorie=fo_categorie*2 WHERE fo_nom='Dupont'; UPDATE 2
9		ROLLBACK; ROLLBACK
10	SELECT fo_nom,fo_categorie FROM fournisseur WHERE fo_nom = 'Dupont'; fo_nom | fo_categorie ------+------- Dupont | 22 Dupont | 22 (2 rows)	

Question 2.3.

Etape	O^2	
0	BEGIN TRANSACTION ; `BEGIN`	
1		BEGIN TRANSACTION ; `BEGIN`
2		

3		SELECT fo_nom,fo_categorie
		FROM fournisseur
		WHERE fo_nom='Dupont' ;
		` fo_nom | fo_categorie`
		`----+-------`
		` Dupont | 20`
		` Dupont | 20`
		`(2 rows)`

4	UPDATE fournisseur SET fo_categorie=fo_categorie+2 WHERE fo_nom='Dupont' ; `UPDATE 2`	
5		UPDATE fournisseur SET fo_categorie=fo_categorie*2 WHERE fo_nom='Dupont' ;
6	COMMIT ; `COMMIT`	 `UPDATE 2`

7		SELECT fo_nom,fo_categorie
		FROM fournisseur
		WHERE fo_nom='Dupont' ;
		` fo_nom | fo_categorie`
		`----+-------`
		` Dupont | 44`
		` Dupont | 44`
		`(2 rows)`

8		ROLLBACK ; `ROLLBACK`

La transaction Terminal 1 est bloquée car elle accède en mise à jour aux même tuples que la transaction Terminal 2.

Question 2.4.

La transaction Terminal 1 est en mode "READ COMMITTED" (par défaut), donc elle a accès aux données mis à jour par l'autre transaction Terminal 1 qui s'est terminée normalement (COMMIT) pendant son exécution.

Question 2.5.

Etape	O^2a	
0	BEGIN TRANSACTION; `BEGIN`	
1		BEGIN TRANSACTION; `BEGIN`
2		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; `SET VARIABLE`

3		SELECT fo_nom,fo_categorie
		FROM fournisseur
		WHERE fo_nom='Dupont';
		` fo_nom | fo_categorie`
		`----+--------`
		` Dupont | 20`
		` Dupont | 20`
		`(2 rows)`

4	UPDATE fournisseur SET fo_categorie=fo_categorie+2 WHERE fo_nom='Dupont'; `UPDATE 2`	
5		UPDATE fournisseur SET fo_categorie=fo_categorie*2 WHERE fo_nom='Dupont';
6	COMMIT; `COMMIT`	 `ERROR: Can't serialize access due to concurrent update`

7		SELECT fo_nom,fo_categorie
		FROM fournisseur
		WHERE fo_nom='Dupont';
		`NOTICE: current transaction is aborted, queries ignored until end of transaction block`
		`*ABORT STATE*`

8		ROLLBACK; `ROLLBACK`

Question 2.6.

Etape	O^{2a} sans l'Etape (3)	
0	BEGIN TRANSACTION; `BEGIN`	
1		BEGIN TRANSACTION; `BEGIN`
2		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE; `SET VARIABLE`
3		
4	UPDATE fournisseur SET fo_categorie=fo_categorie+2 WHERE fo_nom='Dupont'; `UPDATE 2`	
5		UPDATE fournisseur SET fo_categorie=fo_categorie*2 WHERE fo_nom='Dupont';
6	COMMIT; `COMMIT`	 `ERROR: Can't serialize access due to concurrent update`

7		SELECT fo_nom,fo_categorie
		FROM fournisseur
		WHERE fo_nom='Dupont';
		`NOTICE: current transaction is aborted, queries ignored until end of transaction block`
		`*ABORT STATE*`

8		ROLLBACK; `ROLLBACK`

Question 2.7.

Etape	O^2	
0	BEGIN TRANSACTION ; `BEGIN`	
1		BEGIN TRANSACTION ; `BEGIN`
2		

3		SELECT fo_nom,fo_categorie
		FROM fournisseur
		WHERE fo_nom='Dupont'
		FOR UPDATE ;
		` fo_nom | fo_categorie`
		`----+-------`
		` Dupont | 20`
		` Dupont | 20`
		`(2 rows)`

4	UPDATE fournisseur SET fo_categorie=fo_categorie+2 WHERE fo_nom='Dupont' ;	
5		UPDATE fournisseur SET fo_categorie=fo_categorie*2 WHERE fo_nom='Dupont' ; `UPDATE 2`
6		

7		SELECT fo_nom,fo_categorie
		FROM fournisseur
		WHERE fo_nom='Dupont' ;
		` fo_nom | fo_categorie`
		`----+-------`
		` Dupont | 40`
		` Dupont | 40`
		`(2 rows)`

8	 `UPDATE 2`	ROLLBACK ; `ROLLBACK`
6	COMMIT ; `COMMIT`	

Question 2.8.

NON puisque la première transaction est bloquée par `FOR UPDATE` de la deuxième transaction.

Question 2.9.

Il y a un blocage mutuel.

Question 2.10.

Une des transactions (deuxième) est avortée par le système pour rompre le blocage mutuelle : l'autre (première) est débloquée.

Exercice III :

Question 3.1.

Etape	Terminal 1	Terminal 2
1	BEGIN TRANSACTION ; `BEGIN`	
2		DELETE FROM Fournisseur WHERE fo_ville='Athenes' ; `DELETE 1`
3		BEGIN TRANSACTION ; `BEGIN`
4		INSERT INTO Fournisseur VALUES ('F5','Dupont',20,'Paris') ; `INSERT 1`
5		END TRANSACTION ; `COMMIT`
6	UPDATE Fournisseur SET fo_ville='Marseille' WHERE fo_nom='Dupont' ; `UPDATE 2`	

7	SELECT fo_numero,fo_ville
	FROM Fournisseur ;
	`fo_numero | fo_ville`
	`------+------`
	`F1 | Londres`
	`F3 | Paris`
	`F4 | Londres`
	`F2 | Marseille`
	`F5 | Marseille`
	`(5 rows)`	

8	END TRANSACTION ; `COMMIT`	

Question 3.2.

Etape	Terminal 1	Terminal 2
1	BEGIN TRANSACTION ; `BEGIN`	
2	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE ; `SET VARIABLE`	
3		DELETE FROM Fournisseur WHERE fo_ville='Athenes' ; `DELETE 1`
4		BEGIN TRANSACTION ; `BEGIN`
5		INSERT INTO Fournisseur VALUES ('F5','Dupont',20,'Paris') ; `INSERT 1`
6		END TRANSACTION ; `COMMIT`
7	UPDATE Fournisseur SET fo_ville='Marseille' WHERE fo_nom='Dupont' ; `UPDATE 2`	

8	SELECT fo_numero,fo_ville
	FROM Fournisseur ;
	`fo_numero | fo_ville`
	`------+------`
	`F1 | Londres`
	`F3 | Paris`
	`F4 | Londres`
	`F2 | Marseille`
	`F5 | Marseille`
	`(5 rows)`	

9	END TRANSACTION ; `COMMIT`	

Question 3.3.

Etape	Terminal 1	Terminal 2
1	BEGIN TRANSACTION ; BEGIN	
2	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE ; SET VARIABLE	
3	SELECT fo_numero,fo_ville FROM Fournisseur ; fo_numero | fo_ville ------+------ F1 | Londres F2 | Paris F3 | Paris F4 | Londres F5 | Athenes (5 rows)	
4		DELETE FROM Fournisseur WHERE fo_ville='Athenes' ; DELETE 1
5		BEGIN TRANSACTION ; BEGIN
6		INSERT INTO Fournisseur VALUES ('F5','Dupont',20,'Paris') ; INSERT 1
7		END TRANSACTION ; COMMIT
8	UPDATE Fournisseur SET fo_ville='Marseille' WHERE fo_nom='Dupont' ; ERROR: Can't serialize access due to concurrent update	
9	SELECT fo_numero,fo_ville FROM Fournisseur ; NOTICE: current transaction is aborted, queries ignored until end of transaction block *ABORT STATE*	
10	END TRANSACTION ; COMMIT	

Question 3.4.

Etape	Terminal 1	Terminal 2
1	BEGIN TRANSACTION ; BEGIN	
2	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE ; SET VARIABLE	
3		DELETE FROM Fournisseur WHERE fo_ville='Athenes' ; DELETE 1
4	SELECT fo_numero,fo_ville FROM Fournisseur ; fo_numero | fo_ville ------+------ F1 | Londres F2 | Paris F3 | Paris F4 | Londres (4 rows)	
5		BEGIN TRANSACTION ; BEGIN
6		INSERT INTO Fournisseur VALUES ('F5','Dupont',20,'Paris') ; INSERT 1
7		END TRANSACTION ; COMMIT
8	UPDATE Fournisseur SET fo_ville='Marseille' WHERE fo_nom='Dupont' ; UPDATE 1	
9	SELECT fo_numero,fo_ville FROM Fournisseur ; fo_numero | fo_ville ------+------ F1 | Londres F3 | Paris F4 | Londres F2 | Marseille	
10	END TRANSACTION ; COMMIT	