Exercices sur les transactions

Pour réaliser les exercices de ce laboratoire, il faut utiliser deux sessions connectées sur votre base de données.

Préambule

Créer une table `Membres` qui contient les colonnes id de type `NUMBER` (clé primaire), `Nom` de type `VARCHAR2(30)`, Prénom de type `VARCHAR2(25)` et Montant de type `NUMBER(6,2)`.
Y insérer les tuples `(100, 'Dupont', 'Jean', 100.6)`, `(200, 'Dubois', 'Marc', 50)`, `(300, 'Martin', 'Jules', 75.5)` et `(400, 'Tartempion', 'Charles', 25.25)`.

```
CREATE TABLE Membres (
  Id          NUMBER
    CONSTRAINT CpMembres PRIMARY KEY,
  Nom     VARCHAR2(30),
  Prenom      VARCHAR2(25),
  Montant     NUMBER(6,2)
);

INSERT INTO Membres VALUES (100, 'Dupont', 'Jean', 100.6);
INSERT INTO Membres VALUES (200, 'Dubois', 'Marc', 50);
INSERT INTO Membres VALUES (300, 'Martin', 'Jules', 75.5);
INSERT INTO Membres VALUES (400, 'Tartempion', 'Charles', 25.25);
COMMIT;
```

Transaction 1

Imaginer un ensemble de commandes permettant d'illustrer qu'il n'y a pas de lecture impropre, par défaut, dans Oracle.

Session A	Session B
SELECT * FROM Membres WHERE id = 100; => (100, 'Dupont', 'Jean', 100.6)	
	UPDATE Membres SET montant = 15
SELECT * FROM Membres WHERE id = 100; (100, 'Dupont', 'Jean', 100.6)	Commit;

> Le commit n'ayant pas encore été fait dans la session B lors du 2° select de la session A, on obtient encore l'ancienne valeur => pas de lecture impropre par défaut dans Oracle.
> Lecture impropre ou donnée salie : valeur qui n'a pas été confirmée !!!

Transaction 2

Imaginer un ensemble de commandes permettant d'illustrer qu'il n'y a pas de pertes de mises à jour, par défaut, dans Oracle.

Session A	Session B
UPDATE Membres SET montant = montant + 10 WHERE id = 100;	
	UPDATE Membres SET montant = montant + 5 WHERE id = 100
Commit;	attendre
	Commit;

> La commande UPDATE dans la session B ne pourra être effectuée que lorsque le commit aura été exécuté dans la session A => il n'est pas possible de provoquer une perte de mise à jour. => En Oracle, une écriture bloque une écriture.
> Un verrou court dure le temps de l'instruction.
> Un verrou long dure le temps d'une transaction => ici tant qu'on n'a fait ni commit, ni rollback, le verrou reste et la deuxième transaction ne peut continuer !
> En Oracle, une écriture bloque une écriture !
> Ici (à cause de l'instruction de gauche), on a un verrou long de type exclusif.
>
> Exemple : Deux personnes font un retrait de 100€ sur un même compte à 2 distributeurs en même temps, il faut qu'après les 2 retraits, 200€ soient retirés du compte.

> Autre exemple : deux agences de voyage réservent un siège dans le même vol d'avion ... il faut qu'à la fin 2 sièges soient réservés et s'il n'en restait qu'un, il faut que l'une des agence sache qu'elle n'a pas eu de siège !

Transaction 3

Imaginer un ensemble de commandes permettant d'illustrer qu'il est possible d'obtenir des lectures non reproductibles dans Oracle.

Session A	**Session B**
SELECT * FROM Membres WHERE id = 300; => (300, 'Martin', 'Jules', 75.5)	
	UPDATE Membres SET montant = 15 WHERE id = 300;
SELECT * FROM Membres WHERE id = 300; => (300, 'Martin', 'Jules', 75.5)	
	Commit;
SELECT * FROM Membres WHERE id = 300; => (300, 'Martin', 'Jules', 15)	

> Le commit ayant été fait dans la session B lors du 3° select de la session A, on obtient la valeur => les lectures ne sont pas reproductibles par défaut dans Oracle. => importance du commit !!!
> Une lecture non reproductible : dans une même transaction, il se peut que le même select ne donne pas la même valeur !
> Si on fait cet exercice en faisant, dans la session B un insert => dans la session A on a un enregistrement en plus dans le résultat de la requête, on parle alors de référence fantôme

Transaction 4

Imaginer un ensemble de commandes permettant de provoquer une étreinte fatale (deadlock) dans le cadre de deux transactions concurrentes.

Session A	Session B
LOCK TABLE Membres **IN SHARE MODE**; UPDATE Membres SET montant = 25.5 WHERE id = 100;	LOCK TABLE Membres **IN SHARE MODE**; UPDATE Membres SET montant = 35 WHERE id = 200;
Attente à cause de `IN SHARE MODE` de session B	Attente à cause de `IN SHARE MODE` de session A

Les deux processus sont en attente, on a donc un deadlock.
Quand 2 utilisateurs s'attendent mutuellement, une des 2 transactions est arrêtée, c'est la loterie, on ne sait pas dire laquelle le sera !

Ou

Session A	Session B
UPDATE Membres SET montant = 25.5 WHERE id = 100;	UPDATE Membres SET montant = montant-10 WHERE id = 200;
UPDATE Membres SET Montant = montant-10 WHERE id = 200;	UPDATE Membres SET montant = montant-5 WHERE id = 100;
COMMIT ;	COMMIT ;
Attente à cause du verrou long de la première instruction Update de la session B	Attente à cause du verrou long de la première instruction Update de la session A qui n'est pas levé et ne pourra pas l'être puisque la session A attend

> On obtient l'erreur Ora -60 car il détecte le dead lock ...

Transaction 5

Imaginer un ensemble de commandes permettant d'illustrer qu'il est possible d'empêcher les lectures non reproductibles.

Session A	Session B
SET TRANSACTION READ ONLY;	
SELECT Montant FROM Membres WHERE id = 400; => Montant : 25.25	
	UPDATE Membres SET montant = 60 WHERE id = 400;
	Commit;
SELECT Montant FROM Membres WHERE id = 400; => Montant : 25.25	

> 4 niveaux d'isolation :
>
> - Read uncommitted
> - Read committed
> - Repeatable read
> - Serializable
>
> Niveau par défaut d'oracle : read committed

Transaction 6

Effectuer les tests ci-dessous. Aux endroits indiqués par => expliquez la réaction d'Oracle.

SESSION A	SESSION B
UPDATE Membres SET Nom ='xxx' WHERE id = 200;	
Spécifier le type de verrou : Verrou Row Share (RS) Verrou long exclusif en écriture	
	UPDATE Membres SET Nom ='yyy' WHERE id = 200; Ceci implique : un attente
COMMIT; => La modification est validée, le verrou est enlevé	
	COMMIT; => La modification est validée, le verrou est enlevé

Expliquez le résultat de

```
SELECT * FROM Membres WHERE id = 200;
```

> L'Update de la session B attend que le verrou mis par la session A soit enlevé (par un commit ou rollback). L'update de la session B peut alors s'effectuer et va modifier la valeur du champ nom que l'on vient juste de modifier dans la session A
> => Valeur du champ nom = 'yyy'

Transaction 7

Effectuer les tests ci-dessous. Aux endroits indiqués par => expliquez la réaction d'Oracle.

SESSION A	SESSION B
SELECT * FROM Membres FOR UPDATE;	
=> Spécifier le type de verrou Verrou Row Share (RS) Verrou long exclusif en écriture	
	SELECT * FROM Membres FOR UPDATE;
	Oracle attend la fin de la transaction lancée dans la session A

Transaction 8

Effectuer les tests ci-dessous. Aux endroits indiqués par => expliquez la réaction d'Oracle.

SESSION A	SESSION B
SELECT * FROM Membres FOR UPDATE NOWAIT;	
=> Spécifier le type de verrou Verrou Row Share (RS) Verrou long exclusif en écriture	
	SELECT * FROM Membres FOR UPDATE NOWAIT;
	=> On reçoit le message d'erreur : resource busy and acquire with NOWAIT specified or timeout expired