TP N°5
Transactions

Une transaction (ou unité logique de travail) est une séquence de commandes SQL considérée comme unitaire, indivisible.

1 Commandes de contrôle

- Une transaction commence avec la première commande exécutable qui suit la connexion à la base de données, un COMMIT, ou un ROLLBACK.
- Une transaction se termine avec la commande COMMIT, ROLLBACK, ou lors d'une déconnexion.
- Utiliser la commande COMMIT WORK permet de valider la transaction courante : on ne peut plus revenir en arrière.
- Utiliser la commande SAVEPOINT <name> à l'intérieur d'une transaction permet de situer et nommer un point éventuel de retour vers l'état de la base de données au moment de la création de ce point de sauvegarde. Ce retour s'effectue avec la commande

  ROLLBACK WORK TO SAVEPOINT <name>

- Utiliser la commande ROLLBACK sans autre option termine la transaction, annule tous les changements de cette transaction courante, et élimine tous les points de sauvegarde créés lors de cette transaction.

Exercice 1. But: visualiser l'effet de la commande SAVEPOINT <name> :

Transaction T1	
1	*Créer une table* VOLS(no, source, destination):
2	SAVEPOINT p1
3	*Inserer 2 nouveaux tuples*
4	ROLLBACK WORK TO SAVEPOINT p1
5	*Sélectionner tous les tuples de la table* VOLS.

Expliquer le résultat obtenu.

Exercice 2. Donnez l'effet des actions des transactions T1 et T2:

T1	Result T1?	T2	Result T2?
SELECT loc FROM dept WHERE deptno = 20 FOR UPDATE OF loc;			
		UPDATE dept SET loc = 'NEW YORK' WHERE deptno = 20;	
ROLLBACK;			
		COMMIT;	

2 Niveau d'isolation des transactions en ORACLE

Niveau read-commited (par default) Dans le TD precedent nous avons constaté que le verrouillage d'ORACLE est un peu libéral. Le comportement par défaut consiste, pour simplifier, à faire travailler chaque transaction de manière isolée sur une copie de la base(en fait pour être plus précis chaque transaction copie ses modifications dans un espace mémoire dédié, modifications visibles par cette seule transaction). Quand la transaction est validée par un commit alors ses modifications sont appliquées réellement à la base et deviennent visibles à tous. Dans ce mode les lectures ne sont pas verrouillées (pas de verrou de lecture!), mais seules les modifications réalisées par la transaction faisant la lecture, ou alors les modifications réalisées par des transactions validées, seront visibles.

Par contre les écritures disposent d'un verrou. Ainsi si une transaction veut écrire sur une ressource qu'une autre transaction a déjà modifiée, il faut attendre la fin de la transaction ayant pris le verrou en écriture pour continuer. La transaction reste bloquée en attedant le verrou, conduisant parfois à un interblocage. Il est aisé de voir que ce verrouillage ne garantit pas la sérialisabilité!!!

Verrou en lecture Il est possible de verrouiller une ligne en lecture explicitement avec la clause FOR UPDATE. Chaque ligne accéder par le "SELECT" a vant cette clause est verrouillée à la fois en lecture et écriture et donc seule la transaction avec le verrou peut accéder en lecture/écriture sur la ligne ainsi verrouillée. Il s'agit ici d'une restriction très forte. On obtient des exécutions sérialisables pour une raison très simple: il n'existe plus un seul conflit!!

3 Niveau d'isolation: sérialisable

Exercice 3. En supposant L(Oi,Ti) (respectivement E(Oi,Ti)) signifie Lecture (resp. Écriture) de l'objet Oi par la transaction Ti, déterminer si l'exécution suivante est sérialisable. Si oui, donnez une exécution en série équivalente. Justifiez votre réponse en vous basant sur le graphe de précédence:

L(O1, T1), L(O1,T2), E(O2,T3), L(O2,T1), E(O1,T2)