TD-TP DE BASES DE DONNEES
SEANCE 4 : EXPERIMENTATIONS SUR LA CONCURRENCE

Nous allons étudier le comportement d'Oracle en cas d'accès concurrents à la même ressource. Pour cela, nous allons simuler une exécution concurrente à l'aide du petit ensemble de lectures/écritures.

Note : Pour vous connecter sur la VM :

Connection pour avoir accès à la base : oracle / oracle

Connection root pour modifier le clavier en azerty (si besoin) : root / root123

Connection sur le sgbd : sqlplus / as sysdba ← vous ouvre un terminal

Préparation de l'expérience

Créer une table CLI avec deux colonne : NomCli varchar(255) et solde number(6)

Insérer un client nommé 'Joe' (attention au majuscules/minuscules).

Récupérer les 3 fichiers nommés `Solde.sql, Depot.sql et Retrait.sql` sur le serveur.

Ouvrez deux fenêtres SQL*PLUS, chacune sera considérée par Oracle comme pilotant une session différente, en situation de concurrence l'une avec l'autre.

Dans tout ce qui suit, on note $INSTR_i$ l'exécution de l'instruction *INSTR* dans la session *i*. Par exemple $Depot_1$ correspond à l'exécution du fichier `Depot` dans la première fenêtre par la commande `START Depot`. On note de même ROL_i et COM_i l'exécution des commandes `rollback;` et `commit;` dans la session *i*.

Fonctionnement d'Oracle

Q1 : Exécutez les séquences d'instruction décrites ci-dessous. Observez et essayez de comprendre le fonctionnement du verrouillage d'Oracle dans ce mode.

L'utilisateur 1 effectue un retrait sur le compte de 'Joe'. L'utilisateur 2 ne fait que consulter les soldes…

COM_1, COM_2, $Solde_1$, $Solde_2$, $Retrait_1$, $Solde_2$, ROL_1, $Solde_1$, $Solde_2$.

Idem, mais avec un *commit :*

COM_1, COM_2, $Solde_1$, $Solde_2$, $Retrait_1$, $Solde_2$, COM_1, $Solde_1$, $Solde_2$.

L'utilisateur 1 fait un retrait sur le compte de 'Joe', alors que l'utilisateur 2 crédite le compte :

COM_1, COM_2, $Solde_1$, $Solde_2$, $Retrait_1$, $Solde_2$, $Depot_2$, $Solde_1$, $Solde_2$, COM_1, COM_2.

Q2 : ORACLE garantit-il la sérialisabilité des transactions ?

Q3 : Un verrou est-il demandé et/ou placé sur une ligne lors d'une lecture ?

Q4 : Ce comportement correspond-il à un niveau d'isolation décrit dans la norme SQL? Pourquoi ?

Q5 : Cela correspond-il à l'utilisation d'un protocole multi-versions ? Pourquoi ?

Clause FOR UPDATE

Q6 : Modifier le script `Solde.sql` en ajoutant la clause FOR UPDATE. Expérimentez à nouveau les exécutions précédentes.

Q7 : Expliquez ce que vous observez et pour conclure sur la nouvelle stratégie appliquée.

Niveau d'isolation SERIALIZABLE

Q8 : Expérimentez les exécutions précédentes en spécifiant le mode suivant : SET TRANSACTION ISOLATION LEVEL SERIALIZABLE. Attention, cette instruction de changement du degré d'isolation doit être la première de la transaction, et sa portée n'est que la transaction en cours (donc à refaire systématiquement après le COMMIT, autrement une nouvelle transaction démarre en degré d'isolation par défaut).

Q9 : Comparer et expliquer comment Oracle verrouille les données dans ce mode.

Deadlock

Q10 : Donnez une séquence permettant de produire un deadlock. Dans quel(s) niveau(x) d'isolation pourvez-vous créer un deadlock ? Que fait oracle lorsque le deadlock se produit ?