TP Transactions
(observations & discussions)

Nous proposons dans ce TP d'observer le comportement du SGBD Oracle du point de vue des propriétés transactionnelles (ACID).
Ce TP utilisera une relation de comptes bancaires que vous devrez créer. Cette relation comprendra un numéro de compte unique (entier), le nom du propriétaire (chaîne de caractères de taille 10) et un solde (réel).

COMPTES(NC, Nom, Solde)

Sauf mention contraire, les manipulations devront être effectuées en désactivant le mode de validation automatique : vous devez entrer la commande ***set autocommit off;*** dans SQL/Plus. Une instruction **COMMIT** est alors nécessaire après la création de la table COMPTES et pour terminer chaque transaction.

Partie 0 : Mise en place.

Vous trouverez tous les éléments utiles pour vous aider dans le wiki de l'UFR:
https://im2ag-wiki.ujf-grenoble.fr/doku.php?id=environnements:oracle

1. **Connexion au serveur**. La session de travail sous le SGBD Oracle doit impérativement se faire dans une session sur le serveur de bases de données **im2ag-oracle.e.ujf-grenoble.fr** (login: le votre, mot de passe: le votre). Vous pouvez ouvrir une session sur ce serveur à l'intérieur d'une autre session à l'aide de la commande SSH sous Linux, ou bien à l'aide l'outil PuTTY sous Windows.
2. **Choix de l'instance Oracle**. Vous devez maintenant choisir l'instance im2ag pour travailler sous Oracle (par défaut l'instance UFRIMA ne fonctionne pas pour nos TPs). Pour cela entrer la commande **`setenv ORACLE_SID im2ag`**. Ces deux commandes fixent simplement la variable d'environnement ORACLE_SID sur la bonne instance.
3. **Console Oracle**. Lorsque vous êtes dans une session sous im2ag-oracle avec la bonne instance, vous devez encore lancer le client de connexion au SGBD Oracle en tapant la commande **`sqlplus`** (login: le votre, mot de passe perso).Ensuite vous pouvez travailler en tapant des commandes sur la ligne de commande SQL/Plus du client Oracle.
4. **Création du schéma**. Pour créer le schéma vous pouvez vous utiliser le script **bank.sql**
5. **Confort**. Pour gagner en ergonomie penser à faire du copier/coller des commandes entre un éditeur de texte et la ligne de commande Oracle car l'édition sur la ligne de commande est très primitive. Vous pouvez aussi utiliser la commande **start** pour exécuter un script contenant des commandes SQL.

Partie 1 : Atomicité.

Exercice 1 : Annulation de transaction

1. Insérez deux nouveaux comptes ayant pour propriétaire Paul (le numéro de compte et le solde sont à votre choix).
2. Affichez le contenu de la relation COMPTES.
3. Faites un *rollback*
4. Affichez le contenu de la relation COMPTES. Que constatez-vous ?

Exercice 2 : Validation de transaction

1. Dans une nouvelle transaction, insérez deux nouveaux comptes ayant pour propriétaire Pierre.
2. Affichez le contenu de la relation COMPTES.
3. Faites un *commit*.
4. Affichez le contenu de la relation COMPTES.
5. Faites un *rollback*
6. Affichez le contenu de la relation COMPTES. Que constatez-vous ?

Exercice 3 : Points de sauvegarde

1. Dans une nouvelle transaction, insérez un nouveau compte appartenant à Paul.
2. Placez un point de sauvegarde nommé « UnInsert » (*savepoint UnInsert;*)
3. Insérez un nouveau compte appartenant à Paul.
4. Affichez le contenu de la relation COMPTES.
5. Revenez au point de sauvegarde nommé « UnInsert » (*rollback to UnInsert;*)
6. Affichez le contenu de la relation COMPTES.
7. Faites un *rollback*.
8. Affichez le contenu de la relation COMPTES. Que constatez-vous ?

Partie 2 : Isolation.

Cette partie nécessitera d'ouvrir deux sessions (il vous faut deux terminals ouvert) sur le même compte/base de façon à permettre l'accès concurrent à une même base. Ces deux sessions seront nommées S_1 et S_2 dans la suite. Ne pas oublier de désactiver l'autocommit dans chaque session sinon les résultats ne seront pas ceux attendus.

Exercice 4: Niveau d'isolation READ COMMITED (niveau par défaut)

1. Dans une nouvelle transaction de la session S_1, affichez le contenu de la relation COMPTES (requête SELECT).
2. Dans une nouvelle transaction de la session S_2, affichez le contenu de la relation COMPTES.
3. Dans la session S_1, ajoutez 1000 € sur l'un des comptes de Pierre (requête UPDATE).
4. Toujours dans la session S_1, affichez le contenu de la relation COMPTES.
5. Dans la session S_2, affichez le contenu de la relation COMPTES.
   Que constatez-vous ?
6. Validez la transaction de la session S_1.
7. Affichez, dans la session S_2, le contenu de la relation COMPTES. Que constatez-vous ?

Exercice 5 : Verrouillage

Les transactions seront ici en mode READ COMMITTED.

1. Dans une nouvelle transaction de la session S_1, supprimez un (et un seul) compte de Pierre.
2. Ajouter à un compte pour Alain.
3. Affichez, dans une nouvelle transaction de la session S_2, le contenu de la relation COMPTES.
4. Toujours dans la session S_2, modifiez le propriétaire des comptes de Pierre de façon à ce que Paul en soit le nouveau propriétaire. Que constatez-vous ?
5. Validez la transaction de la session S_1. Que constatez-vous dans S_2 ?
6. Affichez, dans la session S_2, le contenu de la relation COMPTES.
   Que constatez-vous ?
7. Validez la transaction de la session S_2.
8. Affichez, dans la session S_1, le contenu de la relation COMPTES.

Que constatez-vous ?

Résumer brièvement la séquence d'événements observée et expliquer comment le SGBD semble évite les modifications concurrentes.

Exercice 6 : Niveau d'isolation SERIALIZABLE.

1. Dans la session S1, supprimez les tuples d'Alain de la relation COMPTES.
2. Validez la transaction de la session S_1.
3. Toujours dans la session S_1, insérez un nouveau compte appartenant à Pierre et ayant un solde de 200 €.
4. Dans une nouvelle transaction de la session S_2, passez en mode SERIALIZABLE (*set transaction isolation level serializable;* dans SQL/Plus).
   Attention : ce mode ne sera actif que pour la transaction en cours.
5. Dans S_2, affichez le contenu de la relation COMPTES.
6. Validez la transaction de la session S_1.
7. Dans S_2, affichez le contenu de la relation COMPTES.
8. Validez la transaction de la session S_2.
9. Dans S_2, affichez le contenu de la relation COMPTES. Que constatez-vous ?

Résumer brièvement les comportements observés avec les deux niveaux d'isolation étudiés dans les exercices précédents. À quel moment les modifications effectuées dans une session deviennent-elles visibles dans une autre session ?

Exercice 7 : Niveau d'isolation SERIALIZABLE (suite).

10. Dans la session S1, ajouter un compte pour jacques.
11. Validez la transaction de la session S_1.
12. Toujours dans la session S_1, modifier le solde du compte de jacques.
13. Dans une nouvelle transaction de la session S_2, passez en mode SERIALIZABLE (*set transaction isolation level serializable;* dans SQL/Plus).
    Attention : ce mode ne sera actif que pour la transaction en cours.
14. Dans S_2, affichez le contenu de la relation COMPTES.
15. Dans S_2, modifiez aussi le compte de jacques avec une autre valeur.
16. Validez la transaction de la session S_1.
17. Validez la transaction de la session S_2.

Résumer brièvement les comportements observés et le problème mis en évidence.

Exercice 8: Deadlock

Vous devez essayer de mettre en mauvaise posture le SGBD en essayant de bloquer le système par un *deadlock* ou *attente mutuelle*. Cette situation apparait lorsqu'une transaction S1 est obligée d'attendre une transaction S2, et que dans le même temps S2 doit attendre S1.

1. Donnez un scénario pour illustrer ce problème.
2. Que constatez-vous ? Est-ce que vous pouvez bloquer Oracle ?