TP n°1 : Transactions (observations)

Nous proposons dans ce TP d'observer le comportement du SGBD Oracle du point de vue des propriétés transactionnelles (ACID)[1].
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), un solde (réel), et une contrainte d'intégrité qui assure un solde positif ou nul (la vérification de cette contrainte aura la possibilité d'être différée, mot clé **`DEFERRABLE`** après la définition de la contrainte).

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.

1. **Connexion**. 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.
   Lorsque vous êtes dans une session sous im2ag-oracle vous devez encore lancer le client de connexion au SGBD Oracle en tapant la commande sqlplus (login: le votre, mot de passe: bd2014, bd2015 ou bd2016… le mot de passe est le même pour tout le monde).
   Ensuite vous pouvez travailler en tapant des commandes sur la ligne de commande SQL/Plus du client Oracle.
2. **Création du schéma**. Pour créer le shéma vous pouvez vous inspirer du contenu du fichier **exemples.sql** qui donnent un ensemble de commandes SQL utiles (création de schéma, ajouter et modifier des tuples, interroger la table).

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.

Pour chaque exercice, décrire brièvement le comportement observé. Enfin, donner en quelques phrases une conclusion globale sur la gestion de l'atomicité.

Exercice 1 : Annulation de transaction

1. Dans une nouvelle transaction[2], 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 ?

---

[1] La propriété de *durabilité* sera étudiée ultérieurement.
[2] Une nouvelle transaction débute (implicitement) après l'établissement d'une connexion (session) avec la base de données ou après la terminaison de la transaction précédente au sein de la même session (après un *commit* ou un *rollback*).

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 4 : 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 : Cohérence

Exercice 1 :

1. Dans une nouvelle transaction, insérez un nouveau compte appartenant à Claude et ayant un solde de 100 €.
2. Insérez un nouveau compte appartenant à Henri et ayant un solde de 200 €.
3. Validez la transaction (*commit*).
4. Passer la vérification de la contrainte sur le solde en mode immédiat (*set constraint X IMMEDIATE;).*
5. Incrémentez le solde du compte de Henri de 50 €.
6. Décrémentez le solde du compte de Claude de 150 €.

Que se passe-t-il ?

7. Affichez le contenu de la relation COMPTES.

Que constatez-vous ?

8. Annulez la transaction (*rollback*).
9. Affichez le contenu de la relation COMPTES.

Que remarquez-vous ?

Refaire ensuite la même expérience avec les trois variantes suivantes. À chaque fois, commencer par détruire les comptes de Claude et Henri et s'assurer que l'on commence bien au sein d'une nouvelle transaction (c'est-à-dire après un appel à *commit* ou *rollback*).
Liste des variantes :

- contrainte en mode *IMMEDIATE* (étape 4) et validation à la fin (étape 8) ;
- contrainte en mode *DEFERRED* (étape 4) et annulation à la fin (étape 8) ;
- contrainte en mode *DEFERRED* (étape 4) et validation à la fin (étape 8).

Décrire brièvement le comportement observé dans chaque cas puis, en quelques phrases, donner une conclusion globale sur la gestion des contraintes de cohérence.

Partie 3 : Isolation.

Cette partie nécessitera d'ouvrir deux connexions à Oracle (dans deux fenêtres) sur le même compte (login) de façon à permettre l'accès concurrent à une même base. Ces deux sessions seront nommées S_1 et S_2. Ne pas oublier de désactiver l'autocommit dans chaque session.

Exercice 1 : Niveau d'isolation READ COMMITED (niveau par défaut)

Il s'agit du niveau d'isolation par défaut d'Oracle.

1. Dans une nouvelle transaction de la session S_1, affichez le contenu de la relation COMPTES.
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.
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 2 : Niveau d'isolation SERIALIZABLE.

1. Dans la session S1, supprimez les tuples de Paul 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 Paul 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[3]).
   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 3 : Verrouillage

1. Dans une nouvelle transaction de la session S_1, supprimez les comptes de Pierre.
2. Affichez, dans une nouvelle transaction de la session S_2, le contenu de la relation COMPTES.
3. Toujours dans la session S_2, modifiez le propriétaire de tous les comptes de façon à ce que Pierre en soit le nouveau propriétaire. Que constatez-vous ?
4. Validez la transaction de la session S_1. Que constatez-vous dans S_2 ?
5. Affichez, dans la session S_2, le contenu de la relation COMPTES.
   Que constatez-vous ?
6. Validez la transaction de la session S_2.
7. Affichez, dans la session S_1, le contenu de la relation COMPTES.

Que constatez-vous ?

Résumer brièvement la séquence d'événements observée et expliquer (en quelques phrases) comment le SGBD gère les modifications concurrentes sur les mêmes données.

Exercice 4 : Interblocage

Vous devez essayer de mettre en mauvaise posture le SGBD en essayant de bloquer le système en le passant dans un état de *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.

Que constatez-vous ? Quel est la stratégie d'Oracle ?

[3] Cette commande ne modifie que le niveau d'isolation de la transaction en cours (et doit être la première instruction de la transaction). Pour modifier le niveau d'isolation de toutes les transactions futures de la session, utiliser : *alter session set transaction isolation level serializable ;*