TRANSACTIONS

Dans les exercices suivants, la base de données `Taxi` sera utilisée.

Lors de ces exercices, il s'agira de comprendre le fonctionnements des transactions concurrentes et l'utilisations de commandes de gestion des transactions.

Pour ces exercices, il sera parfois nécessaire d'ouvrir deux sessions (`sqlplus` dans deux terminaux différents ou `sqlDeveloper`) afin de simuler l'exécutions de deux transactions concurrentes.

1. COMMIT, ROLLBACK

1. Définir une transaction (suite d'opération d'`INSERT`, `UPDATE` ou `DELETE`), valider la la transaction (`COMMIT`), regarder le résultat sur la base de données.

2. Définir une transaction (suite d'opération d'`INSERT`, `UPDATE` ou `DELETE`), annuler la transaction (`ROLLBACK`), regarder le résultat sur la base de données.

2. Contraintes déférées

On souhaite exécuter les deux commandes suivantes au cours d'une même transaction.

```
COMMIT ;
INSERT INTO course
      VALUES ( seq_course.nextval, 2, 200      -- id_chf non existant
               ,'Strasbourg', 'Colmar', 1, 150, 72, CURRENT_DATE);
INSERT INTO chauffeur VALUES (200, 'Kokkotos', 'George');
COMMIT ;
```

Pour exécuter les commandes dans cet ordre, il est nécessaire que la contrainte de clé étrangère soit déférée, c'est à dire que la vérification de la contrainte soit reportée à la fin de la fin de transaction et non au moment où la commande est exécutée.

Redéfinir les contraintes de clé étrangère de la table `course` en `INITIALLY DEFERRED`.
Pour redéfinir une contrainte existante en `INITIALLY DEFERRED`, il est nécessaire de la supprimer et de la recréer.
Pour trouver le nom d'une contrainte nom explicitement nommée, vous pouvez utiliser la table du dictionnaire `USER_CONSTRAINTS`. Les tables du dictionnaires peuvent être obtenues à partir d'une requête sur la table `DICT`.

3. Procédures et transactions

Définir une procédure permettant de supprimer un taxi dont l'identifiant est `id_taxi_p` uniquement si celui ci a subi des réparations. La procédure contiendra les deux commandes suivantes dans l'ordre indiqué :

```
DELETE FROM reparation WHERE id_taxi=id_taxi_p;
DELETE FROM taxi WHERE id_taxi=id_taxi_p;
```

S'il n'y a pas de réparation, la transaction se poursuit comme si la procédure n'avait pas été appelée.
Si la suppression du taxi échoue, la transaction dans sa totalité est annulée.

4. Niveau d'isolation des transactions

Dans ces exercices, deux transactions sont utilisées.

a) A partir de quelle moment une modification (`INSERT`, `UPDATE` ou `DELETE`) effectuée par une transaction `A` est visible par une transaction `B` ?

b) Les lectures non reproductibles et les lectures fantômes sont possibles dans le niveau d'isolation par défaut des transactions `Oracle` (`READ COMMITTED`). On souhaite empêcher les lectures non reproductibles et les lectures fantômes en définissant les transactions au niveau d'isolation `SERIALIZABLE`.
Vérifier que ces anomalies ne se produisent pas pour le niveau d'isolation `SERIALIZABLE`.

5. Interblocage

Provoquer un interblocage à partir de deux transactions concurrentes.

6. Verrouillage explicite des données

a) Vérifier qu'il est possible d'empécher les lectures non reproductibles mais pas les lectures fantômes à l'aide de la clause "`SELECT … FOR UPDATE`".

b) Empêcher les lectures non reproductibles et les lectures fantômes à l'aide en utilisant la commande "`LOCK`".