TP4

Le travail demandé porte sur la base de données du TP1 (Hôtels).
Il faut disposer deux sessions différentes et concurrentes ouvertes sur le même compte Oracle. Pour cela vous pouvez :

- Soit lancer 2 fois le client SQLdeveloper. Puis, dans chaque instance, ouvrir une connexion à la base de données.
- Soit créer une 2ème connexion SQLdeveloper dont les caractéristiques sont identiques à votre 1ère connexion, à l'exception du nom de connexion. Puis ouvrir les 2 connexions.

Attention :

- Dans ces sessions, vous allez exécuter pas à pas des instructions SQL. Pour ce faire, vous devez sélectionner avec la souris l'instruction à exécutée avant de lancer l'exécution en appuyant sur le bouton d'exécution d'un script, car si aucune instruction n'est sélectionnée, toutes les instructions présentes sur la feuille de calcul SQL seront exécutées.
- Vérifier que la validation automatique (autocommit) n'est pas activée dans l'écran :
  Outils -> Préférences -> Base de données -> Avancé
- Avant chaque nouveau test, vous devez réinitialiser vos 2 sessions (fermer les connexions puis les ouvrir à nouveau) ou fermer les transactions courantes (**`commit`** ou **`rollback`**) pour que de nouvelles transactions soient ouvertes (rappel : l'ouverture d'une transaction est implicite lors de l'exécution de la première instruction de la transaction).
- N'allez pas trop vite dans l'exécution des tests, à chaque étape, prenez le temps de bien analyser la situation (données concernées, types d'instruction, résultats d'affichage, …).

1 – Mise en évidence du traitement des écritures sales

Dans la 1ère session, exécuter une instruction SQL qui modifie le nombre d'étoiles d'un hôtel de clef N. Une fois cette instruction exécutée, ne pas terminer la transaction avec une instruction **`commit`**.

Dans la 2ème session, exécuter une instruction SQL qui modifie le nom d'un hôtel dont la clef primaire M est différente de N. Puis exécuter une autre instruction SQL qui cette fois modifie le nom de l'hôtel de clef primaire N.

Que constatez-vous ?

Revenir sur la 1ère session, et exécuter une instruction SQL qui modifie le nombre d'étoiles de l'hôtel de clef primaire M.

Que constatez-vous ?

Maintenant, terminer avec une instruction **`commit`** la transaction initiée dans la 2ème session.

Que constatez-vous ? Quelle conclusion en tirez-vous sur la gestion des écritures sales ?

2 – Mise en évidence du traitement des lectures sales

Dans la 1ère session, exécuter une instruction SQL qui modifie le nombre d'étoiles d'un hôtel de clef primaire N.

Dans la 2ème session, exécuter une instruction SQL qui affiche le nombre d'étoiles de l'hôtel de clef primaire N.

Puis revenir sur la 1ère session pour exécuter la même instruction de lecture.

Que constatez-vous ?

Maintenant, terminer la transaction initiée dans la 1ère session, en exécutant l'instruction **`commit`**. Puis relancer les lectures précédentes dans les 2 sessions.

Que constatez-vous ? Quelle conclusion en tirez-vous sur la gestion des lectures sales ?

3 – Mise en évidence des lectures non reproductibles

Dans la 1ère session, exécuter une instruction SQL qui affiche le nombre d'étoiles d'un hôtel de clef primaire N.

Puis dans la 2ème session, exécuter une instruction SQL qui modifie le nombre d'étoiles de l'hôtel de clef primaire N. Et terminer la transaction avec l'instruction **`commit`**.

Puis revenir sur la 1ère session pour exécuter la même instruction de lecture.

Que constatez-vous ? Quelle conclusion en tirez-vous sur la gestion des lectures ?

4 – Mise en évidence du phénomène des lignes fantômes

Dans la 1ère session, écrire une instruction SQL qui compte le nombre d'hôtels situés à Marseille.

Puis dans la 2ème session, écrire une instruction SQL qui ajoute un nouvel hôtel situé à Marseille. Et terminer la transaction avec l'instruction **`commit`**.

Puis revenir sur la 1ère session pour exécuter la même instruction qui compte le nombre d'hôtels situés à Marseille.

Que constatez-vous ?

Réinitialiser vos 2 sessions, puis refaire le même test avec un autre hôtel, en exécutant au début de votre première session l'instruction : **`set transaction read only ;`**

Que constatez-vous ?

5 – Mise en évidence du phénomène des mises à jour perdues

Tout d'abord, choisir un hôtel existant dans votre table Hôtels dont la clef est C, et prendre connaissance de son nombre d'étoiles.

Puis dans la 1ère session, exécuter le bloc PL/SQL suivant

```
declare
  Nb Integer;
begin
  select nbetoilesho into Nb from hotels where numho = C;
  update hotels set nbetoilesho = Nb +1  where numho = C;
end;
```

Où C doit être remplacé par la valeur de la clef de l'hôtel choisi.

Remarque : Ce bloc PL/SQL permet d'incrémenter le nombre d'étoiles de l'hôtel, tout en séparant la lecture de l'écriture.

Ensuite, dans la 2ème session, exécuter le même bloc PL/SQL.
Puis revenir sur votre 1ère session pour exécuter l'instruction **`commit`**.
Puis aller sur votre 2ème session et exécuter aussi l'instruction **`commit`**.
Enfin prendre connaissance du nombre d'étoiles de l'hôtel qui a été modifié.
Que constatez-vous ?

Réinitialiser vos 2 sessions et refaire le même test mais en démarrant cette fois vos 2 transactions avec l'instruction :

```
set transaction isolation level serializable;
```

Que constatez-vous ?

Est-il envisageable qu'une des 2 transactions ne commence pas par l'instruction :

```
set transaction isolation level serializable;
```

6 – Retour sur le phénomène des fantômes

Ce test suppose qu'il existe au moins 2 hôtels dont le nombre d'étoiles est différent, soient N1 et N2 ces 2 nombres d'étoiles.

Dans votre 1ère session, exécuter l'instruction de modification suivante :

```
update Hotels set NbEtoilesHo = N2 where NbEtoilesHo = N1;
```

Puis dans votre 2ème session, exécuter l'instruction inverse suivante :

```
update Hotels set NbEtoilesHo = N1 where NbEtoilesHo = N2;
```

Enfin, terminer vos 2 transactions par l'instruction **`commit`**.

Que constatez-vous ? (Indication : Quel doit être le résultat de l'exécution en série de ces 2 transactions ?)

Refaire le même test mais en démarrant cette fois vos 2 transactions avec l'instruction :

```
set transaction isolation level serializable;
```

Que constatez-vous ?

7 – Mise en évidence de l'atomicité des instructions SQL

Ce test est indépendant de la gestion de la concurrence, vous utiliserez uniquement la 1ère session. De plus, il suppose l'existence de plusieurs hôtels dans votre table hôtels.

Créer une contrainte qui limite de nombre d'étoiles d'un hôtel à une valeur limite N choisie de sorte que tous vos hôtels existants aient un nombre d'étoiles inférieur ou égal à N-2. Pour cela, exécuter l'instruction ci-dessous où N sera remplacé par la valeur choisie.

```
Alter table Hotels add check( NbEtoilesHo <= N) ;
```

Puis ajouter à la table hôtel un nouvel hôtel dont le nombre d'étoiles sera N.

Ensuite, créer le trigger suivant :

```
Create trigger Trace_Exec
After update of NbEtoilesHo on Hotels For each row
Begin
   Dbms_output.put_line(
   'Modification réussie temporairement sur l''hôtel '||
   :new.NumHo) ;
End ;
/
```

Et configurer l'affichage des sorties en exécutant l'instruction suivante :

```
Set serveroutput on
```

Maintenant afficher le nombre d'étoiles de chaque hôtel, puis exécuter une instruction SQL qui modifie le nombre d'étoiles de chaque hôtel en ajoutant 1, et afficher à nouveau le nombre d'étoiles de chaque hôtel.

Que constatez-vous ?

8 – Gestion de la concurrence dans une contrainte de clef (primaire ou unique)

Dans la 1ère session, exécuter une instruction qui ajoute un nouvel hôtel de clef primaire N. Puis dans la 2ème session, exécuter une instruction qui ajoute aussi un nouvel hôtel de clef primaire M différent de N.

Que constatez-vous ?

Continuer la transaction de la 2ème session en affichant le contenu de la table hôtel puis en exécutant une instruction qui cette fois ajoute un nouvel hôtel ayant la même clef primaire N que l'hôtel ajouté par la transaction de la 1ère session.

Que constatez-vous ?

Revenir sur la 1ère session et exécuter l'instruction **`rollback`**.

Que constatez-vous ?

Quelle conclusion en tirez-vous sur la gestion de la concurrence d'une contrainte de clef ?