Bases de Données Relationnelle 2, TD

Introduction aux transactions

Objectif :

- Comprendre les concepts de base sur les transactions
- Constater directement les notions de base des transactions en interagissant avec un système de BD relationnel

Partie 1 : Révision des concepts de base

1. *Qu'est-ce qu'une transaction?*

   a. Une opération d'écriture dans la base
   b. Une opération suivie d'une opération de lecture
   c. Une séquence d'opérations, lecture ou écriture, terminée par commit ou rollback
   d. La séquence des opérations effectuées par un programme

2. *Dire que deux programmes sont concurrents, c'est dire que*

   a. Ils s'exécutent sur la même machine
   b. Ils communiquent avec le même serveur de données
   c. Is peuvent échanger des messages

3. *J'exécute plusieurs fois de suite le même programme*

   a. J'obtiens toujours la même transaction
   b. J'obtiens toujours la même séquence de transactions
   c. À chaque exécution les transactions peuvent changer
   d. Je n'obtiens jamais la même séquence de transactions

4. *On représente une transaction par une séquence de lecture et d'écriture parce que*

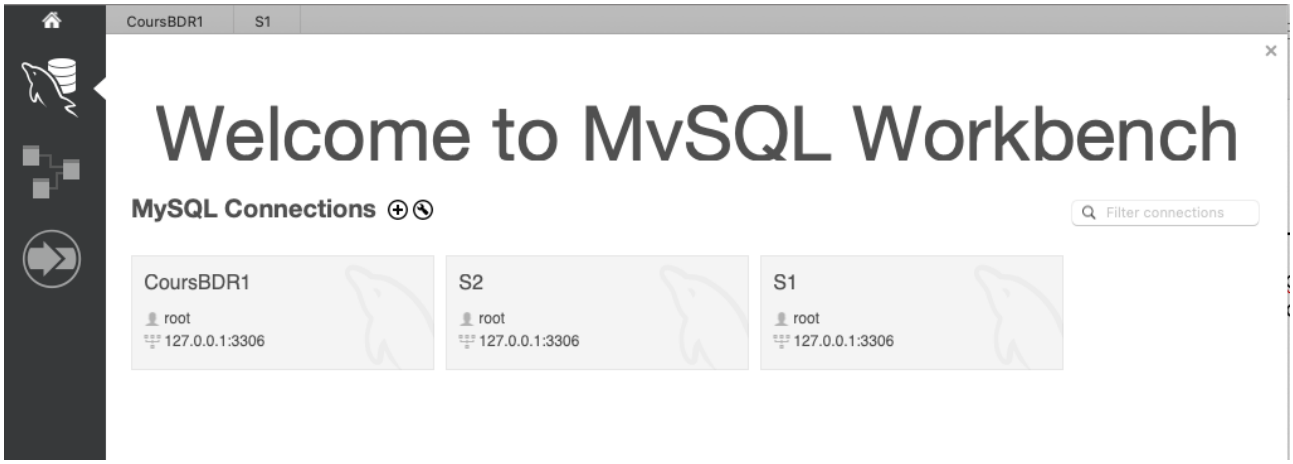
   a. Ce serait trop compliqué de prendre en compte les opérations effectuées par le programme client
   b. Les opérations effectuées par le programme sont inconnues du serveur de données
   c. Connaître les opérations effectuées par le programme ne sert à rien

5. *Les propriétés ACID des transactions sont*

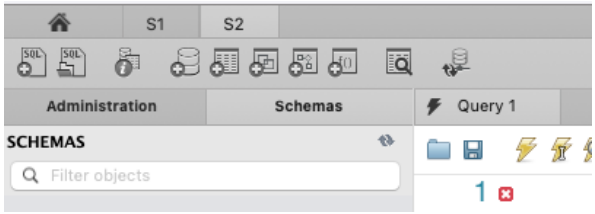
   a. Programmées par le développeur d'application
   b. Garanties par le SGBD
   c. Une situation idéale qui est souvent mise en échec en pratique

Partie 2 : Environnement d'exécution et familiarisation avec la BD

Pour ce TD, vous allez vous connecter à MySQL en utilisant deux sessions : S1 et S2 (vous pouvez les appeler comme vous voulez). Ces deux sessions représentent deux clients (ou programmes) qui accèdent à la même BD.



Vous aurez donc au moins deux onglets ouverts :



Assurez-vous d'avois ces deux sessions ouvertes avant de continuer et d'exécuter sous les deux sessions :

set autocommit = 0;

Nous travaillerons sur la BD ClientSpectacle disponible sur coursenligne. Les opérations effectuées consistent à réserver de places pour le même spectacle.

Important :

Sur cette base on définit la cohérence ainsi: **la somme des places réservées par les clients doit être égale à la somme des places réservées pour les spectacles.**

Dans ce TD, « S1 : » indique qu'il faut exécuter avec la session 1 et « S2 : » avec la session 2.

8. S1 : Exécutez le script de création de la BD
9. S1 : Vérifiez que les données ont été bien insérées
10. S2 : Vérifiez que les données ont été bien insérées
    a. Que remarquez-vous ?

b. Quelle est l'explication ?

11. S1 : Rendez les données de la nouvelle BD disponibles à toutes les sessions

Partie 3 : Les Transactions

Le scénario suivant consiste à réserver, pour le même spectacle, 5 places pour la session 1, et 7 places pour la session 2.

Examinez les données :

12. S1 : SELECT * FROM Client;
13. S1 : SELECT * FROM Spectacle;

La session 1 augmente maintenant le nombre de places réservées :

14. S1 : UPDATE Client SET nb_places_reservees = nb_places_reservees + 5
    WHERE id_client=1;
15. S1 : SELECT * FROM Client;

Notez que la base est ici dans un état instable puisqu'on n'a pas encore diminué le nombre de places libres :

16. S1 : SELECT * FROM Spectacle;

Pour la session 2, la base est dans l'état initial. L'isolation implique que les mises à jour effectuées par la session 1 sont invisibles puisqu'elles ne sont pas encore validées :

17. S2 : SELECT * FROM Client;

Maintenant la session 2 tente d'effectuer la mise à jour du client :

18. S2 : UPDATE Client SET nb_places_reservees = nb_places_reservees + 7
    WHERE id_client=1;
    a. Qu'est-ce que vous observez ?
19. S1 : UPDATE Spectacle SET nb_places_libres = nb_places_libres - 5
    WHERE id_spectacle=1;
20. S1 : commit ;

Réessayez avec la session 2 :

21. S2 : UPDATE Client SET nb_places_reservees = nb_places_reservees + 7
    WHERE id_client=1;
    a. Expliquez pourquoi ça a fonctionné maintenant.
    b. Les mises à jour de la session 1 sont elles maintenant disponibles pour la session 2 ?

Notez que pour l'instant la base est dans un état incohérent puisque 12 places sont réservées par le client, alors que le nombre de places libres a diminué de 5. La seconde session doit décider, soit d'effectuer la mise à jour de Spectacle, soit d'effectuer un rollback. En revanche il et absolument exclu de demander un commit à ce stade, même

si on envisage de mettre à jour Spectacle ultérieurement. Si cette dernière mise à jour échouait, ORACLE ramènerait la base à l'état – incohérent – du dernier commit,

Voici ce qui se passe si on effectue le rollback :

22. S2 : rollback;
23. S2 : SELECT * FROM Client;
    a. Qu'est-ce que vous observez ?

Partie 4 : Quelques questions pour en finir

24. *J'effectue une mise à jour d'un nuplet u, et je n'ai pas encore validé ni annulé. Quelle affirmation est vraie parmi celles ci-dessous ?*

    a. Si je lis u je ne vois pas encore ma mise à jour
    b. Si une autre transaction lit u, elle ne voit pas encore ma mise à jour
    c. Les autres transactions peuvent modifier u

25. *J'effectue un commit, puis je m'aperçois d'une erreur: est-il encore temps de faire un rollback?*
    a. Oui
    b. Non