École Polytechnique de Montréal
Département Génie Informatique et Génie Logiciel
INF3710 – Fichiers et Bases de données

TP 4 – Algèbre relationnelle et Transactions

Objectifs:

- Comprendre les bases théoriques de SQL et effectuer des requêtes en algèbre relationnelle et SQL
- Appliquer les concepts reliés aux transactions

1. Informations générales

Pondération	**5%**
Taille de l'équipe	**2 personnes**

Notez bien:
1. Tout retard dans la remise du TP entraîne automatiquement une pénalité comme discuté dans le plan de cours.
2. Aucun TP ne sera corrigé s'il est soumis par une équipe dont la taille est différente **de deux (2) étudiants** sans l'approbation préalable du chargé de laboratoire. Cette approbation ne sera accordée à une équipe qu'en cas de nombre impair d'étudiants dans le laboratoire. Sinon, la note de zéro sera attribuée aux étudiants concernés.
3. Soumission du TP par **Moodle** uniquement (https://moodle.polymtl.ca). Aucune soumission "hors **Moodle**" ne sera corrigée. La note de zéro sera attribuée aux étudiants concernés.

2. Travail à faire

Partie 1 – Exercices non notés

Connectez-vous à une fenêtre psql.
Créez une base de données TP4

```
CREATE DATABASE tp4;
```

-- connectez-vous à la base de données TP4

```
\c tp4;
```

Q1. Exécutez les commandes suivantes dans psql.

```
create table Compte (username varchar(8), fullname varchar(128),
balance int);
insert into Compte values ('ajones', 'Alice Jones', 82);
insert into Compte values ('sgain', 'Serge Gainsbourg', 10002);
insert into Compte values ('jbirkin', 'Jane Birkin', 5000);
```

Q2. Examinez la structure de la table en utilisant la commande spéciale `\d` :

```
\d compte
select username, fullname, balance from Compte;
select fullname from Compte where balance > 75;
select sum(balance) from Compte;
```

Q3. Nous allons maintenant simuler des transactions concurrentes en ouvrant deux terminaux psql. Nous allons utiliser la couleur bleue et rouge pour les différencier. Ecrivez une transaction dans le terminal T1 qui affiche tous les comptes.

Maintenant, dans le second terminal (rouge), démarrez une transaction et ajoutez un compte pour l'usager 'Charlotte Gainsbourg'.

Q4. Générez une liste de tous les comptes dans le terminal T2. Quelle sortie obtenez-vous ?

Q5. Générez une liste de tous les comptes dans le terminal 1 (bleu). Quelle sortie obtenez-vous ? Inclut-elle Charlotte Gainsbourg ? Pourquoi?

Q6. Validez la transaction dans le terminal T2 rouge et listez l'ensemble des comptes dans les deux terminaux.

Vous voyez maintenant que les deux terminaux voient les mêmes données.

Q7. Dans le terminal 1 bleu, commencez une transaction et ajoutez 5$ dans le compte de Jane Birkin.

Dans le second terminal rouge, commencez une transaction et enlevez 10$ du compte de Jane Birkin.

Q8. Que se passe-t-il dans le terminal T2 rouge ? Pourquoi?

Essayons d'interrompre la transaction dans le terminal T1 (bleu) avec la commande abort :

```
abort;
```

Q9. Que se passe-t-il dans le terminal T2 rouge ?

Q10. Valider la transaction dans le terminal T2. Quel est le solde de Jane Birkin ?

Q11. Effectuons un transfert de 15$ entre Serge Gainsbourg et Charlotte Gainsbourg.

Dans un premier temps, dans le terminal T1 bleu, listez les soldes de tous les comptes. Ensuite commencez une transaction et effectuez le retrait de 15$ dans le terminal T2 rouge.

Q12. Si on regarde tous les soldes dans le premier terminal bleu, est-ce que les soldes ont changé?

Q13. Finissez le transfert en rajoutant 15$ à Charlotte Gainsbourg dans le terminal T2 rouge, puis validez votre transaction. Après chaque commande, listez l'ensembles des soldes dans le terminal T1 bleu pour voir à partir de quand les effets de la seconde transaction deviennent visibles.

Q14. Dans chaque terminal, supprimez le mode autocommit. Créez une nouvelle transaction dans le terminal 1 en ajoutant 15$ à Charlotte Gainsbourg.
Effectuez maintenant la même opération dans T2. Que constatez-vous?
Validez maintenant chaque transaction. Que se passe-t-il?

Répétez l'opération avec la commande suivante dans chaque fenêtre.

```
begin;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
update Compte set balance=balance+15 where username='cgain';
```

Validez maintenant chaque transaction. Que se passe-t-il?

Partie 2 – Algèbre relationnelle et SQL (notée, 44 points)

Soit la base de données suivante :

```
Danseur (DanseurId varchar(5), nom varchar(50), nationalite
varchar(20), age integer)

Spectacle (SpectacleId varchar(5), titre varchar(50), annee integer,
DirArtID varchar(5), categorie varchar(10))
Foreign key (DirArtID) references DirecteurArtistique(DirArtID)

DirecteurArtistique (DirArtID varchar(5), nom varchar(50), nationalite
varchar(20))

Performance (DanseurId varchar(5),SpectacleId varchar(5), role
varchar(50))
Foreign key (DanseurId) references Danseur
Foreign key (SpectacleId) references Spectacle
```

1.1. (*14 points*) Exprimez les requêtes suivantes en algèbre relationnelle (2 points par question). Les symboles d'algèbre sont disponibles sur le fichier Symboles de l'algèbre relationnelle dans le cours 7.

1) Retrouver les détails de tous les spectacles en 2010
2) Retrouver le détail de tous les danseurs qui ne sont pas dans la vingtaine
3) Retrouver le nom de tous les directeurs artistiques Canadiens
4) Retrouver le nom de chaque danseur ainsi que les titres des Spectacles dans lesquels il/elle s'est produit
5) Trouver les noms de tous les danseurs qui ont dansé le rôle du 'cygne' ainsi que l'année du spectacle
6) Retrouver toutes les informations des danseurs du Spectacle 'Opus Cactus' sans opération non nécessaire (indice : vous ne pouvez pas utiliser uniquement un join)
7) Retrouver les titres de tous les spectacles dans lesquels les danseurs Philippe et Kate ont dansé ensemble

1.2. (*30 points*) Exprimez les requêtes suivantes en algèbre relationnelle et SQL (5 points par question, vous n'avez pas besoin de joindre de fichier SQL)

8) Quel est l'âge moyen des danseurs ? Stockez-le dans une colonne nommée AgeMoyen.
9) Quels danseurs (Nom) ont dansé dans au moins un spectacle où la danseuse Lucie Tremblay n'a pas dansé ?
10) Quel est le nombre de spectacles du danseur dont l'id = 1 ? Stockez le résultat dans une colonne nommée nbSpectacle.
11) Affichez une liste des danseurs ainsi que les spectacles (ID) qui leur sont associés s'ils existent, sinon affichez null. L'attribut en commun ne doit pas être répété.
12) Combien de spectacles existent par catégorie ? Stockez le résultat en donnant un nom à la ou les colonnes correspondantes de la relation résultat.
13) Quels danseurs (affichez leurs détails) n'ont participé à aucun spectacle ?

Partie 3 – Transactions (notée, 21 points)

Veuillez noter que vous ne pourrez compléter cette partie qu'après le cours suivant la semaine de relâche.

Soit le code SQL dans le fichier *account.sql* que vous devrez compléter pour les questions suivantes. Précédez chaque question d'un commentaire indiquant le numéro de la question.
Exemple :
-- Q1
-- Q2

1) Soit le tableau correspondant ci-dessous. Complétez *account.sql* avec ces instructions en les précédant de commentaires indiquant la transaction A ou B.

Etape	Session A	Session B

1	`\set AUTCOMMIT 'off';`
	`BEGIN;`
	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`
	`SELECT balance -200 as bal`	

	into balancea FROM Accounts WHERE acctID = 101; SELECT bal FROM balancea;	
2		\set AUTCOMMIT 'off'; BEGIN; SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT balance - 500 as bal into balanceb FROM Accounts WHERE acctID = 101;/ SELECT bal from balanceb;
3	 UPDATE Accounts SET balance = (select bal from balancea) WHERE acctID = 101;	
4		UPDATE Accounts SET balance = (select bal from balanceb) WHERE acctID = 101;
5	SELECT acctID, balance FROM Accounts WHERE acctID = 101; COMMIT;	
6		SELECT acctID, balance FROM Accounts WHERE acctID = 101; COMMIT;

Pour pouvoir exécuter les commandes dans *account.sql*, vous devez ouvrir deux fenêtres psql concurrentes et exécuter chaque étape dans l'ordre indiqué.

a) (*5 points*) Que se passe-t-il quand vous exécutez ces deux transactions concurrentes ? Quel est le problème ? Indiquez la réponse dans un commentaire dans le fichier sql et dans votre rapport.

b) (*5 points*) Comment pourrions-nous nous assurer que les résultats soient cohérents ? Re-générez les données initiales et complétez *account.sql* avec le code SQL approprié en le précédant par des commentaires indiquant à chaque fois la transaction requise tel que montré dans le fichier SQL initial. Indiquez aussi la solution dans un commentaire dans le fichier sql et dans votre rapport.

2) Soit le tableau de transactions suivant. Complétez *accounts.sql* avec le code ci-dessous.

Etape	Session A	Session B
1	\set AUTCOMMIT 'off'; BEGIN; SET TRANSACTION ISOLATION LEVEL READ COMMITTED; --ISOLATION LEVEL REPEATABLE READ; SELECT * FROM Accounts WHERE balance > 500;	
2		\set AUTCOMMIT 'off'; BEGIN; UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101; UPDATE Accounts SET balance = balance + 500 WHERE acctID = 202; SELECT * FROM Accounts; COMMIT;
3	SELECT * FROM Accounts WHERE balance > 500;	

a) (*2 points*) Quel problème constatez-vous ? Indiquez la réponse dans un commentaire dans le fichier sql et dans votre rapport.

b) (*2 points*) Que se passe-t-il si changez le niveau d'isolation de la transaction A à REPEATABLE READ? Indiquez la réponse dans un commentaire dans le fichier sql et dans votre rapport.

c) (*2 points*) Soit le tableau de transactions suivant. Quel problème constatez-vous ? Indiquez la réponse dans un commentaire dans le fichier sql et dans votre rapport.

Etape	Session A	Session B
1	\set AUTCOMMIT 'off'; BEGIN; SET TRANSACTION ISOLATION LEVEL REPEATABLE READ READ ONLY;	
2		\set AUTCOMMIT 'off'; BEGIN; INSERT INTO Accounts (acctID, balance) VALUES (301,3000);
3	SELECT * FROM Accounts WHERE balance > 1000;	
4		INSERT INTO Accounts (acctID, balance) VALUES (302,3000);

5	SELECT * FROM Accounts WHERE balance > 1000; COMMIT;	

3) (*5 points*) En utilisant la table *accounts* et dans le fichier *accounts.sql*, créez un code SQL qui génère un blocage mutuel entre deux transactions

3. Rapport

Votre rapport doit se présenter sous forme de fichier PDF intitulé *matricule1_matricule2_TP2.pdf* contenant :

- La page de présentation disponible sur Moodle, comme page de garde de votre compte rendu ;
- La réponse à toutes les questions de la partie 2. Indiquez clairement leur numéro.
- Des copies d'écran pour chacune des questions de la partie 3 (transactions) ainsi que les réponses aux questions quand cela est demandé. Indiquez clairement le numéro de la question.

4. Modalités de remise

Vous devez soumettre sur Moodle un fichier zip *matricule1_matricule2_TP4.zip* contenant

a. Le fichier *matricule1_matricule2_accounts.sql*
b. Le fichier *matricule1_matricule2_TP4.pdf*