IUT2 Université Grenoble Alpes
Département INFO

R3.07

SQL dans les langages de programmation

Transactions et gestion de la concurrence

Ce TP illustre l'utilisation des transactions sous Postgres.

Nous travaillerons sur le schéma relationnel suivant qui permet de gérer les informations nécessaires dans une Ecole du Ski Français dans une station de ski.

TYPECOURS(numtype, discipline, public, niveau)
MONITEUR(nummono, nom, prenom, adresse, telephone)
COURS(numcours, lieuRV, #numtype, datedeb, datefin, heuredeb, heurefin)
ENSEIGNE (#nummono, #numcours)
ELEVE(#numeleve, nom, prenom, datenaissance, adressestation, mobile)
INSCRIT (#numeleve, #numcours)

Préambule : Création de la base de données et insertion des données

- Lancez un terminal (*Terminal ou Konsole*).
- Sur votre répertoire racine, créez un répertoire R3.07, et dans ce répertoire R3.07 un sous-répertoire ESF.
- Dans ce sous-répertoire ESF, copier les fichiers *create.sql, drop.sql et insert.sql* disponibles dans */users/info/pub/2a/R3.07/esf*
  - *create.sql* : fichier de création des tables,
  - *drop.sql* : fichier de suppression des tables,
  - *insert.sql* : fichier d'insertion des données, à compléter si besoin
- Dans le terminal, connectez-vous sur Postgres par la commande
  psql –h postgres-info –U user base
  user et base doivent être remplacés par votre login
  Le mot de passe sera votre login.
- Exécutez la commande ***\i create.sql*** pour créer les tables de la base de données.
- Si besoin le fichier ***drop.sql*** vous permet de supprimer les tables...
- Exécutez la commande ***\i insert.sql*** pour insérer les données dans la base de données.

Vous noterez que le numéro d'élève, de type serial, est généré automatiquement (voir fichiers create.sql et insert.sql).

Faites un compte-rendu de TP contenant pour chaque question :

- **l'exécution des transactions sous forme de tableau (modèle fourni dans TableauExecutionTransactions.docx),**
- **les réponses données par le SGBD,**
- **vos conclusions sur les observations réalisées.**

Exercice 1 : Transaction, Commit, Rollback

Par défaut, en Postgres, chaque commande utilisateur est exécutée dans sa propre transaction et une validation est lancée implicitement à la fin de la commande (si l'exécution a eu lieu, sinon une annulation est faite). Donc toutes les commandes que vous avez effectuées lors des TPs en BUT1 ont été exécutées dans une transaction. Par exemple, l'insertion suivante constitue implicitement une transaction pour Postgres :

```
insert into MONITEUR
 values(10,'lacroix','franck', 'courchevel','0490584525') ;
```

Pour initier explicitement une transaction, on utilise le mot clé « BEGIN » : toutes les opérations utilisateur après BEGIN seront exécutées dans une transaction unique jusqu'à un COMMIT ou un ROLLBACK explicite.

Vous taperez toutes les commandes dans un fichier *trans1.sql* avant de les exécuter.

1) Commencer une transaction par BEGIN et essayer de faire les manipulations suivantes **Conseils** : afin d'observer le résultat de chaque instruction, NE PAS FAIRE ***\i trans1.sql***, mais faire plutôt des copier/coller instruction par instruction. De plus les copier/coller depuis un fichier pdf sous postgres fonctionnent très mal !!!

```
BEGIN;
insert into MONITEUR values
        (10,'lacroix','franck','courchevel','0490584525');
select * from MONITEUR where nummono = 10 ;
ROLLBACK ;
select * from MONITEUR where nummono = 10 ;
```

Commenter chaque action ! Que fait le serveur Postgres ?

2) Recommencer dans une autre transaction, de la façon suivante :

```
BEGIN;
insert into MONITEUR
   values(10,'lacroix','franck','courchevel','0490584525');
select * from MONITEUR where nummono = 10 ;
COMMIT ;
select * from MONITEUR where nummono = 10 ;
```

Commenter chaque action ! Que fait le serveur Postgres ?

Exercice 2 : Intégrité des données

Essayer les transactions suivantes et commenter. Quel est l'intérêt d'utiliser des transactions ?

1) On insère un nouveau moniteur :

```
BEGIN;
insert into MONITEUR values
        (null,'lapierre','éric,'chamrousse','0490584525');
select * from MONITEUR;
COMMIT;
select * from MONITEUR;
```

2) On veut insérer un nouvel élève et l'inscrire au cours numéro 1.

```
BEGIN;
insert into ELEVE(nom, prenom, datenaissance, adressestation, mobile) values ('gigi', `lebronze', `05/12/1971', `sur le télésiège', `0789889988');
select * from ELEVE;
insert into INSCRIT values (`gigi',1);
COMMIT;
select * from ELEVE;
select * from INSCRIT;
```

Exercice 3 : Transactions concurrentes

Pour bien mettre en évidence le fonctionnement des transactions, il est nécessaire d'avoir des transactions concurrentes. Vous allez donc simuler l'utilisation simultanée de la base de données « esf » par 2 utilisateurs en vous connectant à votre base dans 2 fenêtres différentes. Pour éviter des problèmes de droits, c'est le même utilisateur (*users3a01 par exemple*) qui se connecte 2 fois.
Copier le fichier *trans2.sql* disponible dans */users/info/pub/2a/R3.07/esf*

Attention pour les insertions de dates: pensez à mettre le format européen:
set datestyle to european;

Questions 1 et 2 : Exécuter les requêtes dans l'ordre dans lequel elles sont écrites dans le fichier *trans2.sql* et dans la fenêtre correspondant à chaque utilisateur. Note vos conclusions.

Pour les questions suivantes, c'est à vous d'écrire dans le fichier *trans2.sql* les requêtes à effectuer, en respectant l'ordre chronologique, et en indiquant en commentaires l'utilisateur concerné ainsi que vos conclusions.

Question 3 : Dans deux transactions parallèles, insérer deux moniteurs différents (intercaler les requêtes).

Question 4 : Dans deux transactions parallèles, insérer deux fois le même moniteur. Que se passe-t-il lorsque le premier utilisateur valide sa transaction ? lorsqu'il abandonne sa transaction ? (2 cas à écrire)

Question 5 : Le moniteur Jean-Claude Killy se casse une jambe ! Il faut donc affecter ses cours à un autre moniteur. Dans l'urgence, les cours de ce moniteur sont affectés au moniteur Luc Alphand par l'une des secrétaires, et à la monitrice Marielle Goitschel par une autre secrétaire. Exécuter les requêtes dans l'ordre dans lequel elles sont écrites dans le fichier *trans2.sql*.

Question 6 : En raison des mauvaises conditions météo, l'esf décide de retarder de 1h tous les cours au départ du lieu 'Le Centre'. Exécuter les requêtes dans l'ordre dans lequel elles sont écrites dans le fichier *trans2.sql*.

Conclusion : Dans son mode par défaut (appelé mode READ COMMITED), Postgres n'assure pas la sérialisabilité. Dans la question 6), vous avez vu un exemple de lecture non reproductible. Il existe d'autres modes d'isolation des transactions : les modes REPEATABLE READ et SERIALIZABLE qui gèrent mieux ce type de problèmes.

Exercice 4 : Mode REPEATABLE READ

L'objectif de cet exercice est de tester le mode REPEATABLE READ.
Pour qu'une transaction adopte ce mode d'isolation, vous devrez taper **après** avoir démarré la transaction avec BEGIN, la commande suivante :
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ ;

Reprenez chacune des questions de l'exercice 3 et re-testez chacun des cas afin d'observer les différences entre les 2 modes. Pour chaque cas, notez vos observations dans un fichier transRepeatable.sql

Exercice 5 : Mode SERIALIZABLE

L'objectif de cet exercice est de tester le mode SERIALIZABLE.
Pour qu'une transaction adopte ce mode d'isolation, vous devrez taper **après** avoir démarré la transaction avec BEGIN, la commande suivante :
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

Nous travaillerons ici sur l'exemple (simplifié) du cours avec la relation
COMPTE (numc, client, solde)
Vos instructions SQL ainsi que ce que vous remarquerez dans le TP devront être notées dans un fichier transSerializable.sql

1) Créez la table COMPTE sous Postgres
2) Insérez dans cette table au moins 2 n-uplets:
   - le compte d'Alice, de numéro 1 et de solde 1000,
   - le compte de Bob, de numéro 2 et de solde 1000.
3) L'objectif est ici de faire un virement bancaire de 100 euros du compte de Bob vers celui d'Alice. C'est user1 qui est chargé d'effectuer ce virement. User2 fait des select pour vérifier le solde des 2 comptes après chaque manipulation de user1.
   - Effectuez ces manipulations en mode par défaut READ COMMITTED. Que remarquez-vous ?
   - Effectuez ces manipulations en mode REPEATABLE READ. Que remarquez-vous ?
   - Effectuez ces manipulations en mode SERIALIZABLE. Que remarquez-vous ?

4) Dans cette question, user1 modifie à 4000 le solde du compte d'Alice alors que user2 le modifie en même temps à 5000.
   - Effectuez ces manipulations en mode par défaut READ COMMITTED. Que remarquez-vous ?
   - Effectuez ces manipulations en mode REPEATABLE READ. Que remarquez-vous ?
   - Effectuez ces manipulations en mode SERIALIZABLE. Que remarquez-vous ?

5) Le problème d'interblocage est-il résolu avec le mode SERIALIZABLE ? Testez un cas démontrant votre réponse.

6) Initialisez les valeurs des comptes d'Alice et de Bob. Le compte 1 doit avoir un solde de 2000 et le compte 2 un solde de 1000. On considère 2 transactions concurrentes :
* la 1ère transaction modifie le solde du compte d'Alice en lui ajoutant le solde du compte 2 de Bob.
* la 2ème transaction modifie le solde du compte de Bob en lui affectant la valeur 100; puis elle modifie le solde du compte de Bob en lui ajoutant le solde du compte d'Alice.•

- Quels sont les soldes des comptes d'Alice et de Bob dans le cas d'une exécution séquentielle ?

- Effectuez de manière concurrente les transactions en mode

REPEATABLE READ PUIS SERIALIZABLE. Pensez à faire des requêtes après chaque opération pour voir les soldes des comptes. Que constatez-vous ?

Exercice 6 : Problèmes de concurrence

Vous allez réutiliser la base avengers étudiée au semestre 2. Videz votre base. Créez la base avengers et insérez-y des données en utilisant le fichier :

```
/users/info/pub/1a/R2.06/bases/avengers/create_base.sql
```

1) Illustrez un problème de lecture impropre.

2) Illustrez un problème de perte de mise à jour.

3) Illustrez un problème de lecture non reproductible.

4) Illustrez un problème d'interblocage.

5) Quelles solutions proposez-vous pour résoudre les problèmes rencontrés dans les questions 1, 2, 3 et 4 ? Illustrez vos propositions.