UT/UFR SET/LGI3
Bases de données Avancées - 2019-2020

TP: Transactions

L'objectif de ce TP est de mettre en place des situations de concurrence entre transactions et de comprendre les mécanismes de contrôle de concurrence.

- La BD utilisée pour ce TME est Java DB, la version implémentée par Oracle de la BD open source Derby DB d'Apache
- Voir l'extrait de la documentation (en PJ) de Derby DB sur la gestion des transactions

Préparation du TP

1. Installer Derby

   Configurer les PATH

2. Lancer derby en ligne de commandes avec : $ij
3. Vous connecter à une BD avec la commande ***connect***
   a. **En mode mono utilisateur:**

```
ij) connect 'jdbc:derby:tmpdb;user=tmpuser;create=true';
```

où tmpdb est le nom de la base à laquelle on se connecte.

   b. **En mode serveur :**

```
ij> connect 'jdbc:derby://localhost:1110/igueye' user 'igueye'
password 'igueye';
```

où 1110 est le port d'écoute renseignant dans les configuration du serveur, et igueye est un utilisateur créé avec son mot de passe igueye. Voir section questions fréquentes pour les manip à faire.

Rmq: Vérifier que le serveur DerbyDB est bien démarré en cas d'erreurs.

4. Exécuter les commandes suivantes afin de créer la table Account:

```
autocommit off;
CREATE TABLE Account (acctID INTEGER NOT NULL
PRIMARY KEY,
cname VARCHAR(10) NOT NULL,
balance INTEGER NOT NULL);

DESCRIBE Account;

Select * from Account;
commit;
exit;
```

Lancer deux clients Derby :

- Ouvrir 2 fenêtres de terminal. Le premier terminal sera dénommé par **T1**, le deuxième sera dénommé **T2**.
- Dans **chaque terminal**:

```
autocommit off;
```

Exercice 1: Test commit et rollback

Question 1)

- Dans le terminal **T1** exécuter les instructions suivantes et expliquer le résultat observé:

```
ij> SELECT * FROM Account;
ij> INSERT INTO Account (acctID, cname, balance) VALUES
(101, 'Client A', 1000);
ij> INSERT INTO Account (acctID, cname, balance) VALUES
(202, 'Client B', 2000); ij> SELECT * FROM Account;
ij> ROLLBACK;
ij> SELECT * FROM Account;
```

Question 2)

- Dans le terminal **T1** exécuter les instructions des deux étapes suivantes et expliquer le résultat observé entre ces deux étapes:

Étape 1:

```
ij> INSERT INTO Account (acctID, cname, balance) VALUES
(101, 'Client A', 1000);
ij> INSERT INTO Account (acctID, cname, balance) VALUES
(202, 'Client B', 2000);
ij> SELECT * FROM Account;
ij> exit;
```

Étape 2:

Dans le terminal **T1**, se connecter à nouveau à DerbyDB:

```
autocommit off;
SELECT * FROM Account;
```

Question 3)

Quelle commande faut-il ajouter à l'étape 1 pour que les 2 lignes qui ont été insérées dans la table Account soient affichées à l'étape 2?

Exercices sur la concurrence

Étape d'initialisation:

Lancer Derdy DB en mode serveur
Avant chaque question exécuter les instructions suivantes:

- Finir les transactions précédentes; dans **T1** et **T2** en exécutant:
  ij> commit ;

Remettre la table Account dans l'état initial; dans **T1** exécuter:

```
ij> DELETE FROM Account;
ij> INSERT INTO Account (acctID, cname, balance) VALUES
(101, 'Client A', 1000);
ij> INSERT INTO Account (acctID, cname, balance) VALUES
(202, 'Client B', 2000);
ij> INSERT INTO Account (acctID, cname, balance) VALUES
(303, 'Client C', 3000);
ij> SELECT * FROM Account;
```

Choisir le niveau d'isolation correspondant; dans **T1** et **T2** exécuter:

```
ij> SET isolation <niveau-isolation>;
```

Où <*niveau-isolation*> est parmi les niveaux suivants (voir les niveaux SQL-92 et le Tableau 6, page 78 dans la documentation (PJ : derbydev-extrait-transactions):

- *READ UNCOMMITTED*
- *READ COMMITTED* (par défaut)
- *REPEATABLE READ*
- *SERIALIZABLE*

Opérations:
Le compte du 'Client A' correspondra au **granule A**, le compte du 'Client B' au granule B, le compte du 'Client C' au granule C. Chacune des deux transactions T1 et T2 peut effectuer des opérations parmi les opérations suivantes:

```
L(g) : SELECT * FROM Account WHERE acctID = @g;
E(g, v) : UPDATE Account SET balance = balance + v WHERE acctID=@g;
```

L(g) représente la lecture d'un granule g parmi A, B ou C et **@g** est la valeur de acctID correspondante. E(g,v) représente une écriture du granule g en ajoutant la valeur v (la valeur v peut être positive ou négative). À noter qu'en réalité UPDATE réalise aussi une lecture qu'on va ignorer car UPDATE demande un verrou exclusif et pas de verrou partagé). Par exemple, **L(A)** et **E(B,-100)** correspondent aux instructions suivantes:

```
SELECT * FROM Account WHERE acctID = 101;
UPDATE Account SET balance = balance - 100 WHERE acctID = 202;
```

Pour chaque opération on va noter aussi la transaction qui l'exécute, par exemple **L1(A)** pour la lecture du compte 101 par T1. On va noter également par **C1** et **C2** les opérations commit réalisées par T1 et T2 et par **R1** et **R2** les opérations rollback réalisées par T1 et T2.

Exercice 2: Verrouillage

Pour cet exercice le niveau d'isolation des transactions sera SERIALIZABLE (voir tableau page 83 dans la documentation Derby DB : derbydev-extrait-transactions.pdf):

- Les instructions SELECT demandent des verrous partagés
- UPDATE, DELETE et INSERT demandent des verrous exclusifs

Le verrouillage est strict, les verrous sont relâchés après commit ou rollback (voir la description "Shared Locks" et "Exclusive Locks" pour le niveau SERIALIZABLE, page 81 dans la documentation Derby DB : derbydev-extrait-transactions.pdf).

Pour chacune des questions suivantes:

- tester si les exécutions proposées sont possibles
- écrivez les exécutions effectives obtenues en insérant les demandes de verrouillage et déverrouillage pour chaque transaction (écrire par exemple +VP1(g) et -VX2(g) pour un verrouillage et un déverrouillage du granule g par T1 et T2)
- donnez l'ordre en série équivalent et les valeurs finales dans la table Account.

❖ Première partie: Test du verrouillage

Question 1)

Vérifiez qu'une lecture faite par T1 ne bloque pas une lecture faite par T2

- Exécutez l'étape d'initialisation avec le niveau *SERIALIZABLE*
- L1(A) L2(A) C1 C2

Question 2)

Vérifiez qu'une lecture faite par T1 bloque une écriture faite par T2. *Quelle sera l'instruction de T1 qui va débloquer T2?*

- L1(A) E2(A,100) C1 C2

Question 3)

Vérifiez qu'une écriture faite par T1 bloque une lecture faite par T2.

- E1(A, 100) L2(A) C1 C2

Question 4)

Vérifiez qu'une écriture faite par T1 bloque une écriture faite par T2.

- E1(A, 100) E2(A, 100) C1 C2

Question 5)

On suppose maintenant que T1 et T2 travaillent sur des granules différentes (T1 sur A et T2 sur B par exemple). Reprendre l'exécution de la question 4 précédente en changeant le granule pour T2. T2 est-elle toujours bloquée? Pour quelle raison?

❖ Deuxième partie: Verrouillage en deux phases

Exécutions sans interblocage

Question 6)

- Exécutez l'étape d'initialisation avec le niveau *SERIALIZABLE*
- L1(A) E2(A,-500) E1(A, 100) C1 C2

Question 7)

- Exécutez l'étape d'initialisation avec le niveau *SERIALIZABLE*
- L1(B) E2(A, -500) E1(A,100) C1 C2

Question 8 (question 2.1 du TD)

- Exécutez l'étape d'initialisation avec le niveau *SERIALIZABLE*
- L2(B) E2(B, -70) L2(C) E2(C, 70) L1(A) E1(A,-150) L1(B) E1(B,150) C1 C2

Exécutions avec interblocage

Vérifiez que lorsqu'un interblocage se produit, Derby choisit d'abandonner la transaction qui détient le moins de verrous (en supposant ainsi que ce sera la transaction qui aura effectué le moins de travail avant l'interblocage).

Question 9)

- Exécutez l'étape d'initialisation avec le niveau *SERIALIZABLE*
- E1(A, -150) E2(B, -500) E1(C, 70) E1(B,-500) E2(A, -150) C1 C2

Question 10)

- Exécutez l'étape d'initialisation avec le niveau *SERIALIZABLE*

Donnez une exécution concurrente de T1 et T2 effectuant des opérations sur 2 granules différents, au moins deux de ces opérations étant des lectures. Cette exécution doit produire un interblocage.

Question 11 (question TD)

- Exécutez l'étape d'initialisation avec le niveau *SERIALIZABLE*
- L1(A) E1(A,-150) L1(B) L2(B) E2(B,-70) L2(C) E1(B,150) E2(C,70) C1 C2

Exercice 3: Anomalies et niveaux d'isolation SQL-92

- Lire les diapositives du cours sur les niveaux d'isolation SQL-92 et les anomalies des transactions.

Le niveau d'isolation SERIALIZABLE étant restrictif, l'utilisation des verrous pourrait mettre en attente des transactions dont l'exécution ne pose pas de problème pour la cohérence de la BD. Afin d'augmenter la concurrence et diminuer le temps d'attente d'autres niveaux d'isolation peuvent être utilisés. Cet exercice permet d'étudier les anomalies qui peuvent être rencontrées dans ce cas.

Question 1 (Lecture sale)

Une transaction peut lire des données sales (non validées). Cette anomalie peut être observée en utilisant le niveau *READ UNCOMMITTED*.

a) Pour l'exécution suivante, testez que T2 peut lire le compte du Client A qui est modifié par T1, alors que T1 n'a pas encore validé ses modifications. Que peut-on conclure sur la fiabilité de T2? Existe-t-il des verrous?

* Exécutez l'étape d'initialisation avec le niveau READ UNCOMMITTED * E1(A, 100) L1(A) L2(A) C2 R1 L1(A) C1

b) Testez la même exécution en changeant d'abord le niveau d'isolation à *READ COMMITTED* pendant l'étape d'initialisation. T2 voit-elle toujours l'écriture faite par T1? Existe-t-il des verrous? Quel est leur type et avant quelles opérations sont-ils demandés?

Question 2 (Lecture non reproductible)

Le niveau d'isolation sera *READ COMMITTED*, une transaction pourra lire seulement des données validées (on ne peut donc plus avoir des lectures sales). Il peut cependant y avoir des lectures non reproductibles: la relecture de la même ligne dans la même transaction peut montrer un résultat différent.

a) Pour l'exécution suivante, testez que les deux lectures successives du compte appartenant au Client A par T1 affichent deux valeurs différentes. Quels verrous ont été utilisés? Avant quelles opérations ont-ils été demandés et après quelles opérations sont-ils relâchés?

* Exécutez l'étape d'initialisation avec le niveau READ COMMITTED * L1(A) E2(A, 100) C2 L1(A) C1

b) Testez la même exécution en changeant d'abord le niveau d'isolation à *RS (REPEATABLE READ)* pendant l'étape d'initialisation. T1 voit-elle toujours deux valeurs différentes du compte? Quels verrous ont été utilisés? Avant quelles opérations ont-ils été demandés et après quelles opérations sont-ils relâchés?

Question 3 (Lecture fantôme)

Le niveau d'isolation sera *RS (REPEATABLE READ)*. Ce niveau empêche les lectures sales et non reproductibles. Il peut y avoir cependant le problème lié aux lectures fantômes: pour une transaction, si on relance la même requête à deux instants différents, l'ensemble des lignes résultat aux deux instants peut être différent car de nouveaux enregistrements ont pu être ajoutés/supprimés dans/de la base par une autre transaction, ce qui modifie le résultat. Ceci est différent de la lecture non reproductible dans le sens où les enregistrements lus précédemment n'ont pas changé, mais des enregistrements peuvent être introduits/supprimés du résultat de la requête.

a) Pour l'exécution suivante, testez que les deux lectures successives de la table **Account** effectuées par T1 affichent un ensemble de résultats différent.

* Exécutez l'étape d'initialisation avec le niveau REPEATABLE READ * L1 E2 C2 L1 C1

où **L1** représente la lecture de toute la table Account réalisée par T1:

```
SELECT * FROM Account;
```

E2 représente l'insertion d'une ligne correspondante au compte d'un 'Client D' réalisée par T2:

```
INSERT INTO Account VALUES(404, 'Client D', 4000);
```

b) Testez la même exécution en changeant d'abord le niveau d'isolation à *SERIALIZABLE* pendant l'étape d'initialisation. Le niveau SERIALIZABLE empêche les lectures sales, non reproductibles ainsi que les lectures fantôme.

Exercice 4: Test des niveaux d'isolation SQL-92

Tester les exécutions correspondantes aux Questions 1 à 11 de l'exercice 2 pour les niveaux d'isolation: READ UNCOMMITTED, READ COMMITED et REPEATABLE READ.

Questions fréquentes :

- Quitter derby : exit ;
- Demarrer/Arreter le server :
- `java -jar $DERBY_HOME/lib/derbyrun.jar server start`
- `java -jar $DERBY_HOME/lib/derbyrun.jar server shutdown`

- `Voir aussi startNetworkServer`

- DERBY_HOME : sous UNIX (korn Shell) : /opt/Apache/db-derby-vrsion-bin
- Sous Windows: C:\derby voir/preciser le repertoire d'installation\

- `Pour plus de documentation, voir  en PJ. (Derbyadmin.pdf)`

- `Créer un schéma : ij> create schema testing;`

- `Choisir un schéma : ij> set schema testing;`

- `Créer un utilisateur:`

ij> CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY('derby.user.igueye','igueye');