MAC – Laboratoire 2 Etape 2

Edward Ransome & Michael Spierer

Méthodes d’accès aux données - Prof. Eric Lefrançois

Mode transactionnel



Table des matières

[Table des matières 1](#_Toc499494911)

[Introduction 2](#_Toc499494912)

[Structure du code 2](#_Toc499494913)

[Résultats obtenus 2](#_Toc499494914)

[Analyse 3](#_Toc499494915)

[Conclusion 3](#_Toc499494916)

# Introduction

L’objectif de ce laboratoire est d’étudier les différents modes transactionnels offerts par le moteur InnoDB. Pour ce faire, nous allons utiliser un programme Java exécutant en parallèle des opérations sur une base de données en employant les modes de transactions :

* Read Uncommitted
* Read Committed,
* Repeated Read
* Serializable

Les opérations seront variées (avec ou sans transaction, avec ou sans verrouillage explicite…) et après leur exécution, la cohérence de la base de données sera vérifiée.

# Structure du code

Notre classe TransfertMultiple est une classe qui créé une connexion à la base de données avec un certain niveau d’isolation et le mode autocommit désactivé. Sa méthode « demarrer » permet de lancer un nouveau Thread qui va effectuer un nombre fourni de fois une certaine transaction.

Depuis notre classe principale Main, chaque procédure stockée est testée avec les quatre mode d’isolation. Les résultats sont sauvegardés dans un fichier .csv, ce qui permet d’obtenir tous les résultats avec une seule exécution du code.

# Résultats obtenus

Les résultats obtenus sont fournis par notre code sous la forme d’un fichier .csv, et un exemple d’exécution se situe ci-dessous :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Procédure** | **Niveau d'isolation** | **Nombre d'itérations** | **Temps de traitement [ms]** | **Nombre d'interblocages** | **Résultat cohérent** |
| transferer1 | TRANSACTION\_READ\_UNCOMMITTED | 2000 | 7551 | 1252 | FALSE |
| transferer1 | TRANSACTION\_READ\_COMMITTED | 2000 | 4836 | 899 | FALSE |
| transferer1 | TRANSACTION\_REPEATABLE\_READ | 2000 | 5476 | 1645 | TRUE |
| transferer1 | TRANSACTION\_SERIALIZABLE | 2000 | 6552 | 1148 | TRUE |
| transferer2 | TRANSACTION\_READ\_UNCOMMITTED | 2000 | 5665 | 1444 | FALSE |
| transferer2 | TRANSACTION\_READ\_COMMITTED | 2000 | 6692 | 910 | FALSE |
| transferer2 | TRANSACTION\_REPEATABLE\_READ | 2000 | 7198 | 1610 | TRUE |
| transferer2 | TRANSACTION\_SERIALIZABLE | 2000 | 6566 | 1531 | TRUE |
| transferer3 | TRANSACTION\_READ\_UNCOMMITTED | 2000 | 6751 | 1575 | TRUE |
| transferer3 | TRANSACTION\_READ\_COMMITTED | 2000 | 5828 | 1130 | TRUE |
| transferer3 | TRANSACTION\_REPEATABLE\_READ | 2000 | 6156 | 1194 | TRUE |
| transferer3 | TRANSACTION\_SERIALIZABLE | 2000 | 6162 | 1054 | TRUE |
| transferer4 | TRANSACTION\_READ\_UNCOMMITTED | 2000 | 3664 | 0 | TRUE |
| transferer4 | TRANSACTION\_READ\_COMMITTED | 2000 | 5389 | 0 | TRUE |
| transferer4 | TRANSACTION\_REPEATABLE\_READ | 2000 | 4616 | 0 | TRUE |
| transferer4 | TRANSACTION\_SERIALIZABLE | 2000 | 3604 | 0 | TRUE |

# Analyse

On remarque avec ces données obtenues sur 2000 itérations que les interblocages sont présents et nombreux sur toutes les procédures avec tous les modes d’isolation excepté la procédure « transferer4 ». Cette procédure utilise une méthode conçue exprès pour éviter l’interblocage.

Pour les autres procédures, c’est le niveau d’isolation Read Committed qui apporte le moins d’interblocages. En revanche, Repeated Read en apporte le plus.

Les temps d’exécution les plus rapides sont ceux de la procédure stockée « transferer4 », ce qui n’est pas étonnant, car elle n’a aucun interblocage. Par contre, c’est le niveau d’isolation « Serializable » qui est le plus rapide, ce qui est étonnant, car c’est celui qui effectue le plus de verrouillages et traitements avant d’être effectué et celui qui, d’après le cours, est le moins efficace[[1]](#footnote-1).

Il est important de noter qu’en cas d’interblocage, celui-ci est détecté et on n’attend pas le temps de timeout par défaut de 30 secondes. En effet, on obtient l’exception suivante :

*« com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException: Deadlock found when trying to get lock; try restarting transaction »*

Le temps perdu lors des interblocages est donc beaucoup plus petit qu’on pourrait s’imaginer.

Au niveau de la cohérence, seuls les niveaux d’isolation Read Committed et Read Uncommitted peuvent mener à une incohérence. C’est uniquement dans ces deux niveaux d’isolation que les anomalies de type « lectures sales », « lectures non-répétables » et « fantômes » peuvent avoir lieu. Les autres cas ne mènent pas à une incohérence de la base de données. Cette incohérence peut être évitée avec le verrouillage explicite des lignes par le développeur (« tranferer3 » et « transferer4 »).

# Conclusion

Ce laboratoire nous a permis d’implémenter les niveaux d’isolation du moteur InnoDB manuellement plutôt que d’utiliser le mode par défaut fourni par MySQL. Il est important de noter que, par défaut, MySQL n’empêche pas l’interblocage avec le moteur InnoDB.

Nous avons pu voir l’impact des différents modes de transactions et nous avons pu empêcher tout interblocage avec une méthode étudiée en cours.

1. Polycopié cours MAC Octobre 2017, Gestion des transactions – Part II, Page 26, « Moins efficace sera le système… » [↑](#footnote-ref-1)