Transactions MySQL – Verrous

I► Dans cet article, nous allons étudier le principe des transactions et des verrous MySQL.

Une **transaction** permet de rendre **atomique (ACID)** un processus de plusieurs requêtes. le processus est exécuté intégralement ou pas du tout, mais jamais partiellement !

- **Atomicité** :Toutes les requêtes de la transaction sont exécutées avec succès ou toutes annulées.
- **Cohérence** : Si la base de données est dans un état cohérent au début d'une transaction, elle le restera à la fin de la transaction.
- **Isolation** : les requêtes d'une transaction n'affectent pas les autres transactions.
- **Durabilité** : au COMMIT, le système s'assure de l'intégrabilité et de la cohérence des données(aucunes pertes)

? Exemple : cas du virement bancaire

Mme Wayne souhaite acheter un IPhone en payant par CB. La banque reçoit l'ordre de paiement du vendeur concernant la commande de Mme Wayne. Elle passe donc les requêtes comme suit :

InnoDB est par défaut en mode AutoCommit, cela veut dire que chaque requête est traitée de manière indépendante. Si l'on souhaite jouer plusieurs requêtes et vérifier leur intégrité, il faudra désactiver le mode autocommit.

? Optimisation des requêtes et Verrous

Par défaut, une requête lorsqu'elle est envoyée par le client est mise en attente dans une file jusqu'à ce qu'elle soit exécutée. Les requêtes de modification sont prioritaires par rapport aux simples requêtes de sélection. Le système bloque la table ou la ligne grâce aux principes du verrou implicite. La granularité du verrou implicite peut être la ligne seulement (moteur InnoDB) ou la table entière (moteur MyISAM).

Le client peut lui interagir avec le système en posant un verrou de type explicite. Il existe 2 types de verrous explicites (READ et WRITE)

Un verrou posé en lecture seule (READ) bloque la table en modification pour tout le monde y compris le client qui a posé le verrou :
exemple : **LOCK TABLE MaTable READ;**

Un verrou posé en écriture (WRITE) bloque la table en modification pour tout le monde sauf le client qui a posé le verrou, lui peut donc modifier la table :
exemple : **LOCK TABLE MaTable WRITE;**

Pour déposer les verrous, il suffit d'exécuter : **UNLOCK TABLES;**

Exemple :

```
LOCK TABLE MaTableEnLecture WRITE, MaTableEnEcriture WRITE;
INSERT INTO MaTableEnLecture WRITE SELECT * FROM MaTableEnEcriture;
TRUNCATE MaTableEnEcriture;
UNLOCK TABLES;
```

Afin d'optimiser les requêtes MySQL, le client peut donner un ordre de priorité à l'exécution des ses requêtes

INSERT DELAYED into MaTable (col1,col2) VALUES (x,y);
DELAYED pour les requêtes de type INSERT ou REPLACE redonne instantanément la main au client et place la requête dans un tampon mémoire qui n'exécutera les requêtes que lorsque toutes

les autres requêtes ‘normales’ auront été exécutées.

INSERT LOW_PRIORITY into MaTable (col1,col2) VALUES (x,y);
LOW_PRIORITY ne fonctionne que pour les requêtes de type INSERT | UPDATE | REPLACE | DELETE et pas pour le SELECT. Les requêtes de modification perdent leur priorité naturelle …

SELECT HIGH_PRIORITY * FROM MaTable;
HIGH_PRIORITY fonctionne également avec le SELECT.
Les requêtes gagnent en priorité …

Michel BOCCIOLESI