UNIVERSITÉ CÔTE D'AZUR

MIAGE RÉSEAU DES MIAGE DE FRANCE Nice

Projet de recherche

Système de Gestion de Base de Données

MySQL

Nicolas Meyer - Olivier Gaudard - Christophe Santamaria - Cedric Bontemps

Tuteur : Gregory GALLI / Gabriel MOPOLO
Diplôme : Master 1 MIAGE
Université de Nice/Sophia Antipolis
Année : 2020/2021

TABLE DES MATIÈRES

I. IDENTIFICATION DU SGBD

Notre groupe a choisi de s'occuper de la recherche détaillée du système de base de données relationnelles (ou DBMS - Database Management System) MySQL.

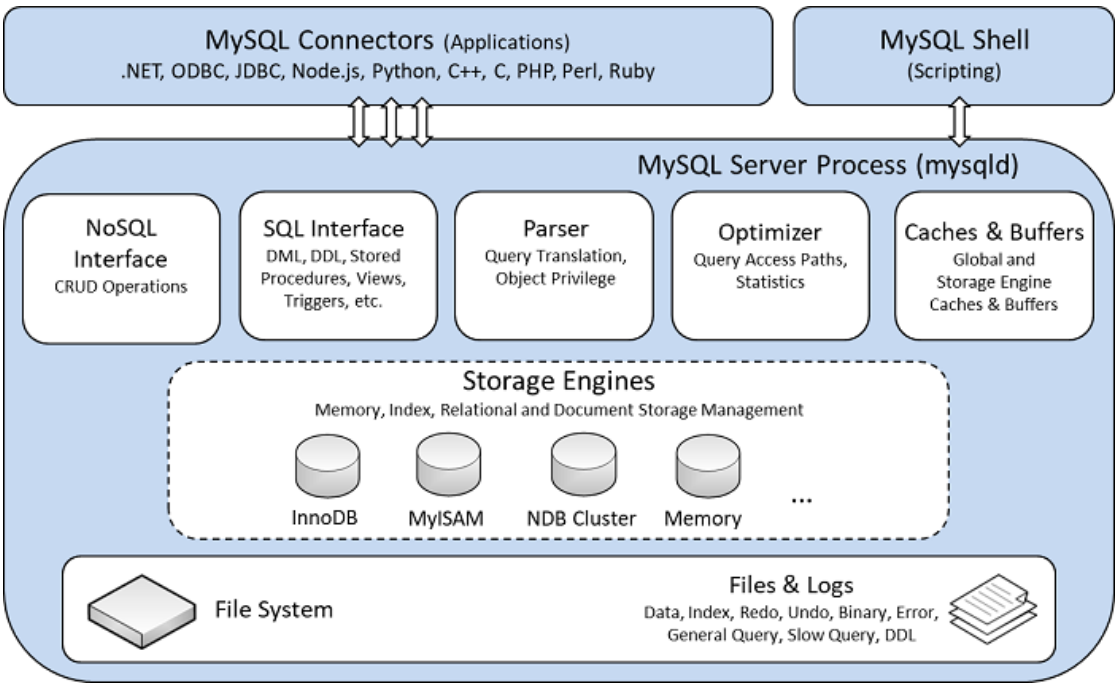
Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde.

La première version de MySQL est apparue le 23 mai 1995, fondée par David Axmark, Allan Larsson et Michael Widenius, deux suédois et un filandais. Il a d'abord été créé pour un usage personnel à partir de mSQL en s'appuyant sur le langage de bas niveau ISAM, jugé trop lent et trop rigide. Ils ont créé une nouvelle interface SQL en gardant la même API que mSQL.

En 2010, MySQL AB est racheté consécutivement par Sun Microsystems et Oracle Corporation. Ce qui permet à Oracle de posséder deux des gros systèmes de gestion de base données autrefois concurrentes.
Les entreprises comme Google, Twitter, Wikipedia, Yahoo, Youtube, Uber... utilisent MySQL pour certaines de leurs applications.

II. ARCHITECTURE FONCTIONNELLES

A. Dessin architecture



B. Caches mémoires, rôles et processus

MySQL alloue des buffers (ou “tampons” en français) et des caches pour améliorer les performances des opérations de base de données. La configuration par défaut est conçue pour permettre à un serveur MySQL de démarrer sur une machine virtuelle disposant d'environ 512 Mo de RAM.

Il est possible d’améliorer les performances de MySQL en augmentant les valeurs de certaines variables système liées au cache et à la mémoire tampon.
Il est également possible de modifier la configuration par défaut pour exécuter MySQL sur des systèmes avec une mémoire limitée.

Exemple de gestion de cache :

- **InnoDB Buffer Pool Optimization**
  InnoDB maintient une zone de stockage appelée le “buffer pool” (ou “pool de mémoire tampon” en français) pour mettre en cache les données et les index en mémoire. Connaître le fonctionnement du pool tampon InnoDB et en tirer parti pour conserver en mémoire les données fréquemment consultées est un aspect important de l'optimisation de MySQL.

- **MyISAM Key Cache**
  Pour minimiser les entrées/sorties sur disque, le moteur de stockage MyISAM exploite une stratégie utilisée par de nombreux systèmes de gestion de bases de données.

C. Gestion des connexions

MySQL permet la création de comptes qui autorisent les utilisateurs à se connecter au serveur et à accéder aux données gérées par le serveur. La fonction principale du système de privilèges MySQL est d'authentifier un utilisateur qui se connecte à partir d'un hôte donné et d'associer à cet utilisateur des privilèges sur une base de données tels que SELECT, INSERT, UPDATE et DELETE. Une fonctionnalité supplémentaire inclut la possibilité d'accorder des privilèges pour les opérations administratives.

Pour contrôler quels utilisateurs peuvent se connecter, chaque compte peut se voir attribuer des informations d'authentification telles qu'un mot de passe. L'interface utilisateur des comptes MySQL consiste en des instructions SQL telles que CREATE USER, GRANT et REVOKE.

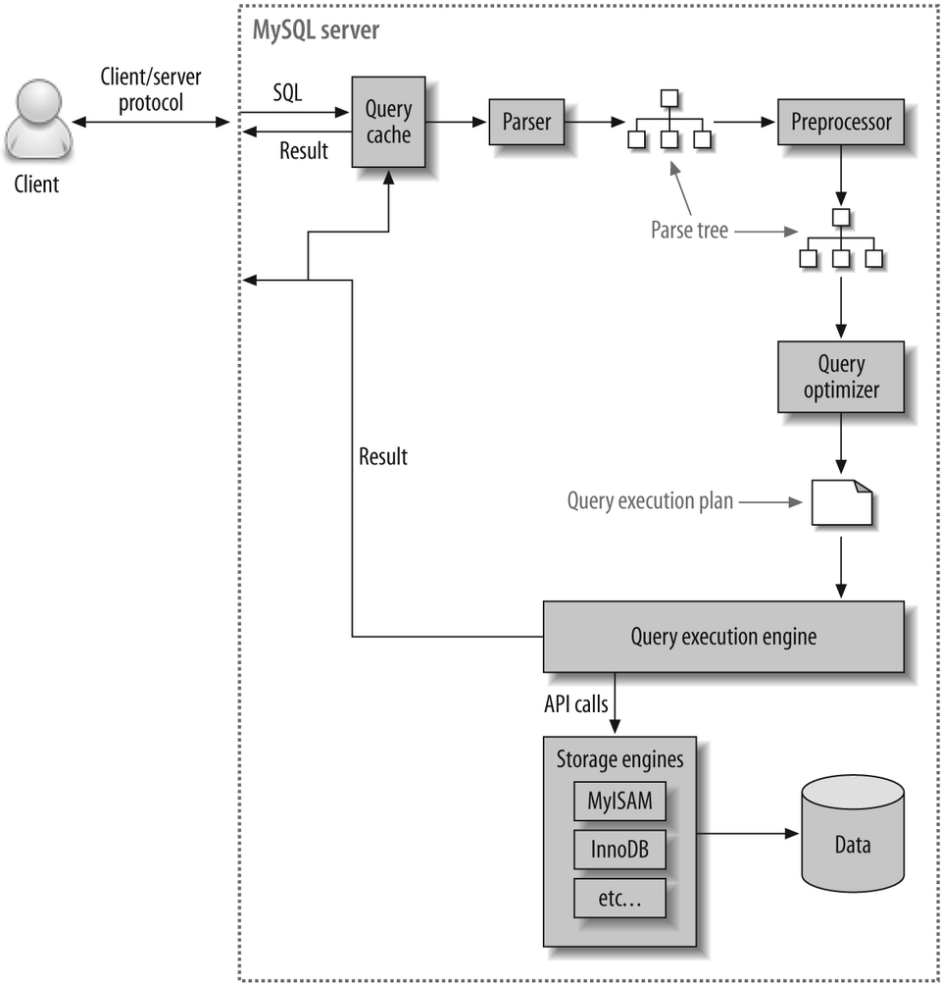
Le système de privilèges MySQL garantit que tous les utilisateurs ne peuvent effectuer que les opérations qui leur sont autorisées.

D. Processus d’exécution des requêtes

Voici le processus théorique d’une exécution d’une requête SQL de l’envoi au serveur d’une requête par un utilisateur (ou d'un client si on parle d’un logiciel) et la réception de la donnée ou le statut de la requête.

1. Le client envoie l'instruction SQL au serveur.
2. Le serveur vérifie le cache des requêtes. S'il y a une correspondance, il renvoie le résultat stocké dans le cache. Sinon, il transmet l'instruction SQL à l'étape suivante.
3. Le serveur analyse, pré-traite et optimise le SQL dans un plan d'exécution de la requête.
4. Le moteur d'exécution des requêtes exécute le plan en effectuant des appels à l'API du moteur de stockage.
5. Le serveur envoie le résultat au client.

Seulement, rien ne vaut un schéma pour représenter graphiquement le processus d'une requête SQL plus en détail.



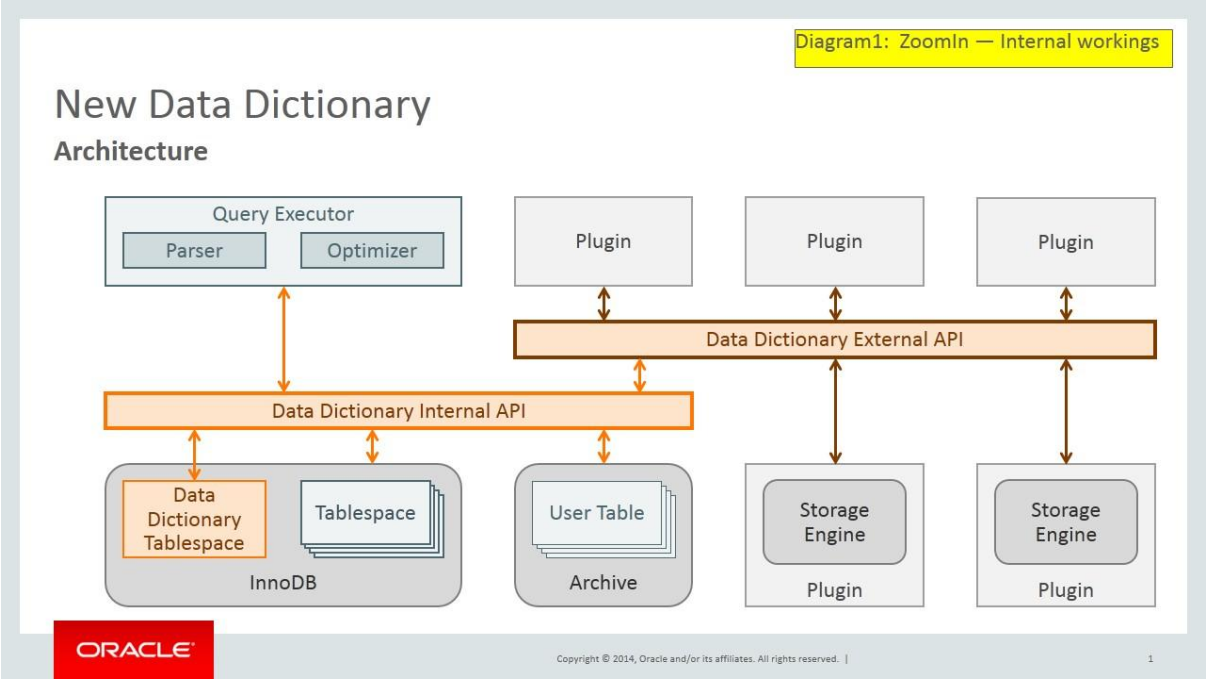
III. DICTIONNAIRE DE DONNÉES

MySQL Server intègre un dictionnaire de données transactionnel qui stocke des informations sur les objets de la base de données. Dans les versions précédentes de MySQL, les données du dictionnaire étaient stockées dans des fichiers de métadonnées, des tables non transactionnelles et des dictionnaires de données spécifiques au moteur de stockage.

Le cache des objets du dictionnaire est un cache global partagé qui stocke en mémoire les objets du dictionnaire de données précédemment accédés. À l'instar des autres mécanismes de cache utilisés par MySQL, le cache des objets du dictionnaire utilise une stratégie d'éviction basée sur LRU pour évincer de la mémoire les objets les moins récemment utilisés.

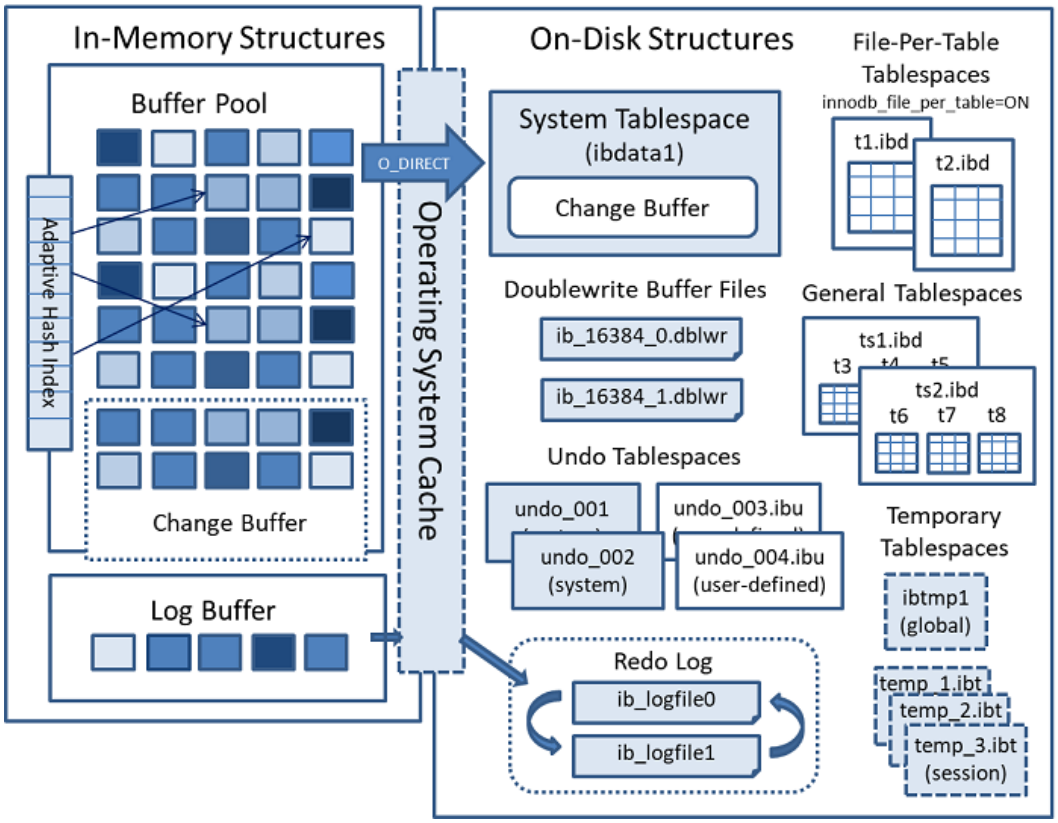
Le cache objet du dictionnaire comprend des partitions de cache qui stockent différents types d'objets. Certaines limites de taille de partition de cache sont configurables, tandis que d'autres sont codées en dur.

Voici un schéma du dictionnaire de donnée de MySQL après la refonte du système de dictionnaire de donnée en MySQL 8.0



IV. ORGANISATION PHYSIQUE DU SGBD

A. Schémas



B. Types de fichiers

Sur **MySQL**, il existe tout un tas d'extension de fichier afin d'aider au fonctionnement du SGBD. Voici la liste non-exhaustive des types de fichiers présents dans l'infrastructure MySQL :

***.frm** : Il s'agit de l'extension de fichier qui contient le schéma ou la définition de la table.
***.myd** : Il s'agit de l'extension du fichier qui contient les données de la table MyISAM.
***.myi** : Il s'agit de l'extension du fichier qui contient les index de la table MyISAM.
my.cnf : C'est un fichier de configuration de base de données MySQL. Il s'agit du fichier de configuration principal du serveur MySQL.
db.opt : Chaque fois qu'une base de données est créée ou modifiée à l'aide des commandes MySQL, les caractéristiques de la base de données sont stockées dans un fichier texte, "db.opt".
***.ibd** : Ce sont les fichiers qui stockent les données et l'index des tables MySQL InnoDB. Ce type de fichier est créé ou utilisé par le logiciel MySQL InnoDB et lui est associé.
***.sock** : Toutes les connexions à la base de données MySQL sont gérées par un fichier spécial appelé fichier socket. Ce fichier de socket, à savoir mysqld.sock, est créé automatiquement par le service MySQL, ce qui facilite la communication entre les différents processus.
.pid : L'ID de processus du serveur MySQL est écrit dans ce fichier. La valeur par défaut sera le nom d'hôte du serveur MySQL.
***.db** : Ce sont les fichiers avec les extensions «.db» qui stockent les données et les index du moteur de stockage BerkeleyDB.

***.log** : Il existe plusieurs fichiers de type .log ; A savoir les erreurs systèmes, les erreurs des requêtes, des erreurs binaires qui sont relatives à la création de données ou de table orchestré par MySQL et les “slow query” qui sont présentes pour améliorer les performances.
***.index** : Pour surveiller les fichiers journaux binaires utilisés, un fichier d'index binaire est créé, contenant les noms de tous les fichiers journaux binaires.
.TMD : Il s'agit du fichier de base de données intermédiaire créé par le serveur MySQL créé lors des opérations de réparation. Ce fichier contient des informations sur les récupérations de bases de données.
***.TRG et *.TRN** : Les fichiers TRG sont des fichiers de paramétrage qui sont déclenchés et les fichiers TRN sont des fichiers d'espace de noms également déclenchés. Dans le serveur MySQL, chaque fois que des triggers (ou “déclencheur” en français) sont définis, les définitions sont stockées dans ces fichiers.
***.ARZ, *.ARM et .ARN** : Les fichiers de données de table et de métadonnées de table ont respectivement les extensions .ARZ et .ARM. Le fichier .ARN est le fichier d'optimisation pendant le processus d'optimisation.

V. ORGANISATION LOGIQUE DES DONNÉES

A. Données des tables

INFORMATION_SCHEMA permet d'accéder aux métadonnées de la base de données, des informations sur le serveur MySQL telles que le nom d'une base de données ou d'une table, le type de données d'une colonne ou les privilèges d'accès. D'autres termes parfois utilisés pour ces informations sont dictionnaire de données et catalogue système.
C’est une base de données dans chaque instance de MySQL. La base de données **INFORMATION_SCHEMA** contient plusieurs tables en lecture seule. Il s'agit en fait de vues, et non de tables de base. Aucun fichier ne leur est donc associé, et il est impossible de définir de déclencheurs sur elles. De plus, il n'y a pas de répertoire de base de données portant ce nom.

B. Données des indexes

Les index sont utilisés pour trouver rapidement les lignes ayant des valeurs de colonnes spécifiques. **Sans index, MySQL doit commencer par la première ligne et parcourir toute la table pour trouver les lignes pertinentes**. Plus la table est grande, plus cela coûte cher. Si la table possède un index pour les colonnes en question, MySQL peut rapidement déterminer la position à rechercher au milieu du fichier de données sans avoir à regarder toutes les données. C'est beaucoup plus rapide que de lire chaque ligne séquentiellement.

La plupart des index MySQL sont stockés dans des **B-trees**. Il existe cependant une exception : Les index sur les types de données spatiales utilisent des arbres R qui équivaut aux tables MEMORY qui supportent également les index de hachage. InnoDB utilise des listes inversées pour les index FULLTEXT.

C. Données temporaire

MySQL dispose d'une fonctionnalité permettant de créer une table spéciale appelée Table temporaire qui nous permet de conserver des données temporaires. Nous pouvons réutiliser cette table plusieurs fois dans une session particulière. Elle est disponible dans MySQL pour l'utilisateur à partir de la version 3.23, et plus, donc si nous utilisons une version plus ancienne, cette table ne peut pas être utilisée. Cette table est visible et accessible uniquement pour la session en cours. MySQL supprime automatiquement cette table tant que la session en cours est fermée ou que l'utilisateur termine la connexion. Nous pouvons également utiliser la commande **DROP TABLE** pour supprimer explicitement cette table lorsque l'utilisateur ne l'utilisera pas.

VI. GESTION DE LA CONCURRENCE D'ACCÈS

A. Transaction

MySQL supporte les transactions locales (au sein d'une session client donnée) par le biais d'instructions telles que **SET autocommit**, **START TRANSACTION**, **COMMIT** et **ROLLBACK**.

B. Support de propriétés ACID

Pour rappel, le modèle ACID est un ensemble de principes de conception de bases de données qui mettent l'accent sur les aspects de la fiabilité. MySQL inclut des composants, tels que le moteur de stockage InnoDB, qui adhèrent étroitement au modèle ACID afin que les données ne soient pas corrompues et que les résultats ne soient pas faussés par des conditions exceptionnelles telles que les pannes logicielles et les dysfonctionnements matériels.

1. Atomicité

L'aspect atomicité du modèle ACID concerne principalement les transactions InnoDB. Les fonctionnalités MySQL connexes comprennent :

- Le paramètre autocommit.
- L'instruction COMMIT.
- L'instruction ROLLBACK.

2. Cohérence

L'aspect cohérence du modèle ACID implique principalement le traitement interne d'InnoDB pour protéger les données contre les pannes. Les fonctionnalités MySQL connexes comprennent :

- Le tampon de double écriture InnoDB.
- La reprise après incident InnoDB.

3. Isolation

L'aspect isolation du modèle ACID concerne principalement les transactions InnoDB, en particulier le niveau d'isolation qui s'applique à chaque transaction. Les fonctionnalités MySQL connexes comprennent :

- Le paramètre autocommit.
- Les niveaux d'isolation des transactions et l'instruction SET TRANSACTION.
- Les détails de bas niveau du verrouillage InnoDB. Les détails peuvent être consultés dans les tables INFORMATION_SCHEMA et dans les tables data_locks et data_lock_waits

4. Durabilité

L'aspect durabilité du modèle ACID implique que les fonctionnalités du logiciel MySQL interagissent avec une configuration matérielle particulière déterminée par l'administrateur SGBD. Les fonctionnalités MySQL associées incluent :

- Le tampon de double écriture InnoDB.
- La variable innodb_flush_log_at_trx_commit.
- La variable sync_binlog.
- La variable innodb_file_per_table.
- Le tampon d'écriture dans un périphérique de stockage, tel qu'un disque dur, un SSD ou une matrice RAID.
- Un cache alimenté par batterie dans un périphérique de stockage.
- Le système d'exploitation utilisé pour exécuter MySQL.
- Une alimentation sans coupure protégeant l'alimentation électrique de tous les serveurs informatiques et périphériques de stockage qui exécutent les serveurs MySQL et stockent les données MySQL.
- Une stratégie de sauvegarde déterminée par l'administrateur SGBD, telle que la fréquence et les types de sauvegardes, ainsi que les périodes de conservation des sauvegardes.
- Pour les applications de données distribuées ou hébergées, les caractéristiques particulières des centres de données où se trouve le matériel des serveurs MySQL, et les connexions réseau entre les centres de données.

C. Technique de gestion de la concurrence d'accès

L'absence de la gestion de la concurrence d'accès peut entraîner de nombreux problèmes d'incohérence dus aux opérations des différents utilisateurs en simultanés. C'est pourquoi il existe un certain nombre de techniques afin de garantir une cohérence des ressources de la base de données. La gestion de la concurrence d'accès se traduit par l'utilisation de verrous, permettant de restreindre ou de bloquer l'accès à des ressources en simultanés par différents utilisateurs. Cela peut s'appliquer sur des tables entières, des blocs ou n-uplets.

Il existe plusieurs types de verrous :

- Le verrou READ permet de laisser la lecture des données de l'élément verrouillé aux autres sessions mais bloque la modification des données.

- Le verrou WRITE permet de bloquer la lecture et la modification de l'élément verrouillé aux autres sessions

Lorsqu'une session obtient un verrou, cette dernière peut accéder uniquement aux éléments verrouillés par l'utilisation du nom de l'élément ou de l'alias affecté. Si il s'agit d'un verrou de lecture, les autres sessions peuvent également acquérir un verrou de lecture sur ce même élément. Par contre, s' il s'agit d'un verrou d'écriture, les autres sessions ne pourront pas accéder à cet élément tant que le verrou existe.

D. Sérialisation

En plus de stocker les métadonnées sur les objets de la base de données dans le dictionnaire de données, MySQL les stocke sous forme sérialisée. Ces données sont appelées informations sérialisées du dictionnaire (SDI).

Les informations de dictionnaire sérialisées (SDI) sont présentes dans tous les fichiers de tablespace InnoDB, à l'exception des fichiers de tablepace temporaires et de tablespace d'annulation. Les enregistrements SDI d'un fichier tablespace InnoDB décrivent uniquement les objets de la table et du tablespace contenus dans le tablespace.

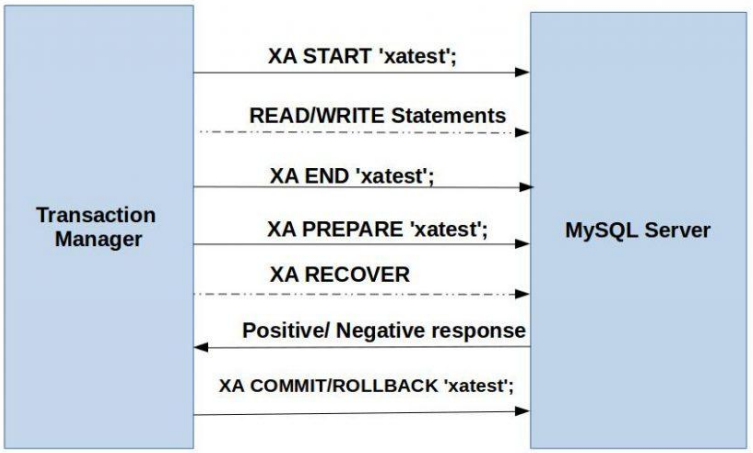
VII. GESTION DES TRANSACTIONS DISTRIBUÉES

XA, pour "eXtended Architecture", est une norme créée par The Open Group pour le traitement des transactions distribuées.

A. Architecture

XA utilise un protocole de validation en deux phases, la première phase étant la demande de validation suivie de la validation proprement dite. Une fois que les différentes branches de la transaction globale ont terminé l'exécution, le protocole de validation en deux phases entre en jeu :

- Dans la première phase, le gestionnaire de transactions envoie un message de préparation à la validation à toutes les branches impliquées dans la transaction globale. Avant que le gestionnaire de ressources ne reconnaisse qu'il est prêt pour le commit, il enregistre les résultats des actions dans un stockage stable pour effectuer le commit réel dans la deuxième phase.
- Dans la deuxième phase, le gestionnaire de transactions reçoit une réponse affirmative de toutes les branches impliquées, alors elles sont notifiées de commettre. Cependant, si l'une des branches répond négativement, toutes les branches sont notifiées pour faire marche arrière.



B. Douze règles de Date

Les 12 règles de Codd permettent de définir un système de gestion de base de données comme relationnel. Ces règles ont été énoncées par Edgar Codd en 1985, on peut aujourd'hui les considérer comme incomplètes car l'intégralité de nos SGBD relationnels moderne les respecte à différents niveau, mais elles restent les racines du modèle relationnel et chaque nouvel acteur du marché doit les suivre pour que son produit soit considéré comme relationnel.

Ces règles décrivent les fonctionnalités de bases que doivent avoir les SGBD pour être considérées comme relationnelles. Parmis elles ont pour trouver :

- La représentation de la donnée d'une seule manière dans les champs de table.
- Un syntaxe linéaire qui peut être utilisé dans des programmes. Le langage doit fournir au minimum : définition de valeur de recherche supplémentaire ; manipulation des données ; traitement de gestion des transaction ; ajout d'option et de contraintes.
- Toutes les vues doivent pouvoir être mises à jour et modifiées via le système.

C. Garantie de l'atomicité, la cohérence et la durabilité

XA aborde le problème de la préservation des propriétés ACID au sein d'une transaction unique. Les ressources elles-mêmes peuvent être d'autres serveurs MySQL, ou même des technologies de base de données différentes. **Le standard pour XA décrit l'interaction entre le gestionnaire de transactions global et le gestionnaire de ressources local**.

Une transaction globale implique plusieurs actions qui sont transactionnelles en elles-mêmes, mais qui doivent toutes se terminer avec succès en tant que groupe, ou être annulées en tant que groupe. En substance, cela étend les propriétés ACID "à un niveau

supérieur" de sorte que plusieurs transactions ACID peuvent être exécutées de concert en tant que composants d'une opération globale qui possède également des propriétés ACID.

D. Traitement des pannes

Récupération après une corruption de données ou une défaillance de disque

Si la base de données est corrompue ou si une défaillance du disque se produit, il faut effectuer la récupération en utilisant une sauvegarde. En cas de corruption, on doit d'abord trouver une sauvegarde qui n'est pas corrompue. Après avoir restauré la sauvegarde de base, une récupération ponctuelle doit être effectuée à partir des fichiers journaux binaires à l'aide de **mysqlbinlog** et **mysql** pour restaurer les modifications survenues après la sauvegarde.

Dans certains cas de corruption de la base de données, il suffit de vider, d'abandonner et de recréer une ou quelques tables corrompues.

Récupération ponctuelle

Pour restaurer une base de données InnoDB au présent à partir du moment où la sauvegarde physique a été effectuée, il faut exécuter le serveur MySQL avec la journalisation binaire activée, avant même de prendre la sauvegarde.

Récupération après un crash InnoDB

Pour récupérer d'un crash inattendu du serveur MySQL, il suffit de redémarrer le serveur MySQL. InnoDB vérifie automatiquement les journaux et effectue un roll-forward de la base de données jusqu'au moment présent. InnoDB annule automatiquement les transactions non engagées qui étaient présentes au moment du crash.

La récupération d'un crash InnoDB se fait en plusieurs étapes :

- **Découverte du tablespace**
- **Application du journal des opérations**
- **Retour en arrière des transactions incomplètes**
- **Fusion du tampon de changement**
- **Purge**

Découverte de tablespaces pendant la récupération après un crash

Si, pendant la récupération, InnoDB rencontre des redo logs écrits depuis le dernier checkpoint, les redo logs doivent être appliqués aux tablespaces affectés. Le processus qui identifie les tablespaces affectés pendant la récupération est appelé découverte des tablespaces.

VIII. GESTION DE LA REPRISE SUR PANNE

A. Techniques d'annulation

Pour les annulations, la technique la plus simple reste le Rollback. **ROLLBACK** annule la transaction en cours, en annulant ses modifications. Cette technique ne marche que si l'on a pas validé la transaction.

Le fait d'utiliser **BEGIN** avant de lancer la requête permet également de rollback à tout moment.

Si on a déjà validé la transaction (par commit ou en quittant le client en ligne de commande), on peut restaurer les données à partir de la dernière sauvegarde par exemple.

Si on configure la journalisation binaire sur MySQL, on peut restaurer la base de données à n'importe quel point précédent pour lequel on a encore un instantané et un journal bin.

B. Journalisation

Pour restaurer une base de données InnoDB au présent à partir du moment où la sauvegarde physique a été effectuée, vous devez exécuter le serveur MySQL avec la journalisation binaire activée, avant même de prendre la sauvegarde. Pour réaliser une récupération ponctuelle après la restauration d'une sauvegarde, vous pouvez appliquer les modifications du journal binaire qui se sont produites après l'exécution de la sauvegarde.

C. Sauvegardes

Réalisation d'une sauvegarde à chaud avec MySQL Enterprise Backup

Les clients de MySQL Enterprise Edition peuvent utiliser le produit MySQL Enterprise Backup pour effectuer des sauvegardes physiques d'instances entières ou de bases de données ou tables sélectionnées, ou les deux. Ce produit comprend des fonctionnalités de sauvegardes incrémentielles et compressées. La sauvegarde des fichiers physiques de la base de données rend la restauration beaucoup plus rapide que les techniques logiques telles que la commande mysqldump. Les tables InnoDB sont copiées à l'aide d'un mécanisme de sauvegarde à chaud. (Idéalement, les tables InnoDB devraient représenter une majorité substantielle des données). Les tables provenant d'autres moteurs de stockage sont copiées à l'aide d'un mécanisme de sauvegarde à chaud.

Réalisation de sauvegardes avec mysqldump

Le programme mysqldump peut effectuer des sauvegardes. Il peut sauvegarder toutes sortes de tables.
Pour les tables InnoDB, il est possible d'effectuer une sauvegarde en ligne qui ne prend aucun verrou sur les tables en utilisant l'option --single-transaction de mysqldump.

Réalisation de sauvegardes en copiant des fichiers de tables

Les tables MyISAM peuvent être sauvegardées en copiant les fichiers de table (fichiers *.MYD, *.MYI, et fichiers associés *.sdi). Pour obtenir une sauvegarde cohérente, il faut arrêter le serveur ou verrouiller et vider les tables concernées.
Il est possible de créer une sauvegarde binaire en copiant les fichiers de table, tant que le serveur ne met rien à jour.

Réalisation de sauvegardes de fichiers de texte délimité

Pour créer un fichier texte contenant les données d'une table, Il est possible d'utiliser ***SELECT * INTO OUTFILE 'file_name' FROM tbl_name***.
Le fichier est créé sur l'hôte du serveur MySQL, et non sur l'hôte du client. Pour cette instruction, le fichier de sortie ne peut pas déjà exister car le fait de permettre l'écrasement des fichiers constitue un risque pour la sécurité.

Pour recharger un fichier de données au format texte délimité, il faut utiliser LOAD DATA ou mysqlimport.

Réalisation de sauvegardes incrémentielles en activant le journal binaire

MySQL prend en charge les sauvegardes incrémentielles en utilisant le journal binaire. Les fichiers journaux binaires fournissent les informations pour répliquer les modifications apportées à la base de données après le moment où la sauvegarde a été effectuée. Par conséquent, pour permettre à un serveur d'être restauré à un point dans le temps, la journalisation binaire doit être activée sur celui-ci, ce qui est le paramètre par défaut pour MySQL 8.0.

Réalisation de sauvegardes en utilisant des répliques

Si des problèmes de performances sont constatés avec un serveur lorsqu'il y a des sauvegardes, une stratégie qui peut configurer la réplication et effectuer les sauvegardes sur la réplique plutôt que sur la source.

D. Gestion de la reprise à chaud

La commande mysqlbackup, qui fait partie du composant MySQL Enterprise Backup, permet de sauvegarder une instance MySQL en cours d'exécution, y compris les tables InnoDB, en perturbant le moins possible les opérations tout en produisant un instantané cohérent de la base de données. Lorsque mysqlbackup copie les tables InnoDB, les lectures et les écritures dans les tables InnoDB peuvent continuer. MySQL Enterprise Backup peut également créer des fichiers de sauvegarde compressés et sauvegarder des sous-ensembles de tables et de bases de données. En conjonction avec le journal binaire MySQL, les utilisateurs peuvent effectuer une restauration ponctuelle. MySQL Enterprise Backup fait partie de l'abonnement MySQL Enterprise.

E. Gestion de la reprise à froid

Pour la reprise à froid, il faut arrêter le serveur MySQL, effectuer une sauvegarde physique qui consiste en tous les fichiers utilisés par InnoDB pour gérer ses tables. et enfin, il faut respecter la procédure suivante :

- Arrêt lent du serveur MySQL.
- Copie de tous les fichiers de données InnoDB.
- Copie de tous les fichiers journaux InnoDB.
- Copie de tous les fichiers de configuration my.cnf.

IX. TECHNIQUES D'INDEXATION

La meilleure façon d'améliorer les performances des opérations ***SELECT*** est de créer des index sur une ou plusieurs des colonnes qui sont testées dans la requête. Les entrées de l'index agissent comme des pointeurs vers les lignes de la table, ce qui permet à la requête de déterminer rapidement quelles lignes correspondent à une condition de la clause ***WHERE*** et de récupérer les valeurs des autres colonnes pour ces lignes.

Par contre, les index inutiles gaspillent de l'espace et font perdre du temps à MySQL pour déterminer les index à utiliser. Les index augmentent également le coût des insertions, mises à jour et suppressions, car chaque index doit être mis à jour. Il faut trouver le bon équilibre pour obtenir des requêtes rapides en utilisant un ensemble optimal d'index.

A. Indexation B-Tree

Un index B-tree peut être utilisé pour les comparaisons de colonnes dans les expressions qui utilisent les opérateurs **=, >, >=, <, <=** ou ***BETWEEN***. L'index peut également être utilisé pour les comparaisons ***LIKE*** si l'argument de ***LIKE*** est une chaîne constante qui ne commence pas par un caractère générique. Par exemple, les instructions ***SELECT*** suivantes utilisent des index :

```
1. SELECT * FROM tbl_name WHERE key_col LIKE 'Nicolas%';
2. SELECT * FROM tbl_name WHERE key_col LIKE 'Nic%_las%';
```

Dans la première instruction, seules les lignes avec 'Nicolas'<= key_col < 'Nicolat' sont prises en compte. Dans la deuxième instruction, seules les lignes avec 'Nic' <= key_col < 'Nid' sont prises en compte.

Si on utilise... ***LIKE*** '%string%' et que la longueur de la chaîne est supérieure à trois caractères, MySQL utilise l'algorithme **Turbo Boyer-Moore** pour initialiser le modèle de la chaîne, puis utilise ce modèle pour effectuer la recherche plus rapidement.

Une recherche utilisant col_name ***IS NULL*** utilise des index si col_name est indexé.

Tout index qui ne couvre pas tous les niveaux ***AND*** de la clause ***WHERE*** n'est pas utilisé pour optimiser la requête. En d'autres termes, pour pouvoir utiliser un index, un préfixe de l'index doit être utilisé dans chaque groupe ***AND***.

Exemple d'utilisation d'index :

```
... WHERE index_part1=1 AND index_part2=2 AND autre_colonne=3

    /* index = 1 OU index = 2 */
... WHERE index=1 OR A=10 AND index=2

    /* optimisé comme "index_part1='hello'" */
... WHERE index_part1='hello' AND index_part3=5

    /* Peut utiliser l'index sur l'index1 mais pas sur l'index2 ou l'index3 */
... WHERE index1=1 AND index2=2 OR index1=3 AND index3=3 ;
```

Parfois, MySQL n'utilise pas d'index, même s'il en existe un. Cela se produit notamment lorsque l'optimiseur estime que l'utilisation de l'index obligerait MySQL à accéder à un pourcentage très élevé des lignes de la table. Cependant, si une telle requête utilise ***LIMIT*** pour ne récupérer qu'une partie des lignes, MySQL utilise quand même un index, car il peut trouver beaucoup plus rapidement les quelques lignes à retourner dans le résultat.

B. Indexation R-Tree

SPATIAL INDEX crée un index R-tree. Pour les moteurs de stockage qui prennent en charge l'indexation non spatiale des colonnes spatiales, le moteur crée un index B-tree. Un index B-tree sur des valeurs spatiales est utile pour les recherches de valeurs exactes, mais pas pour les balayages de plage.

X. OPTIMISATION DE REQUÊTES

Les techniques d'optimisation des requêtes s'appliquent aux constructions telles que ***CREATE TABLE...AS SELECT, INSERT INTO...SELECT*** et les clauses ***WHERE*** des instructions ***DELETE***. Ces instructions ont des considérations supplémentaires en matière de performances car elles combinent des opérations d'écriture avec des opérations de requête orientées lecture.

Pour rendre plus rapide une requête ***SELECT ... WHERE*** lente, la première chose à vérifier est de savoir si on peut ajouter un index. Configurer des index sur les colonnes utilisées dans la clause ***WHERE***, afin d'accélérer l'évaluation, le filtrage et la récupération finale des résultats.

Les index sont particulièrement importants pour les requêtes qui font référence à différentes tables, en utilisant des fonctionnalités telles que les jointures et les clés étrangères. Il est possible d'utiliser l'instruction ***EXPLAIN*** pour déterminer quels index sont utilisés pour un ***SELECT***. On peut également réduire au minimum le nombre d'analyses de tables complètes dans les requêtes, en particulier pour les grandes tables. Il est préférable d'éviter de

transformer la requête d'une manière qui la rende difficile à comprendre, surtout si l'optimiseur effectue automatiquement certaines de ces transformations.

Enfin, il est possible d'ajuster la taille et les propriétés des zones de mémoire que MySQL utilise pour la mise en cache. Grâce à une utilisation efficace du pool tampon InnoDB, du cache de clés MyISAM et du cache de requêtes MySQL, les requêtes répétées s'exécutent plus rapidement car les résultats sont récupérés en mémoire la deuxième fois et les fois suivantes.
Gérer les problèmes de verrouillage, où la vitesse de la requête peut être affectée par d'autres sessions accédant aux tables en même temps.

XI. PROJET THÉMATIQUE INDEXATION ET RECHERCHE

https://github.com/MiageNicoMeyer/sgbd-galli-grp3

XII. BIBLIOGRAPHIE

1. https://dev.mysql.com/doc/refman/8.0/en/ (référence globale)

2. https://fr.wikipedia.org/wiki/MySQL (partie I)

3. https://dev.mysql.com/doc/refman/8.0/en/pluggable-storage-overview.html (partie II, A - Schéma)

4. https://dev.mysql.com/doc/refman/8.0/en/buffering-caching.html (partie II - B)

5. https://mysqlserverteam.com/a-preview-on-lab-release-with-new-data-dictionary-in-mysql/ (partie III)

6. https://www.geeksforgeeks.org/mysql-database-files/ (partie IV, B - Schéma)

7. http://man.hubwiz.com/docset/MySQL.docset/Contents/Resources/Documents/innodb-storage-engine.html (partie IV, B)

8. https://mysqlserverteam.com/improvements-to-xa-support-in-mysql-5-7/ (partie VII)

9. https://www-igm.univ-mlv.fr/~lecroq/string/node15.html (Algorithme Turbo Boyer-Moore)

10. https://mysqlserverteam.com/geographic-indexes-in-innodb/ (R-tree MySQL)