Projet de recherche sur les SGBD du marché

2020 - 2021

Rédigé par :

Sacha Bydon
Stéphane Désiré
Allan Duvois
Samuel Maugard
Sabrina Mercuri

Encadré par :

M. Galli
M. Mopolo

SOMMAIRE

1. Identification du SGBD

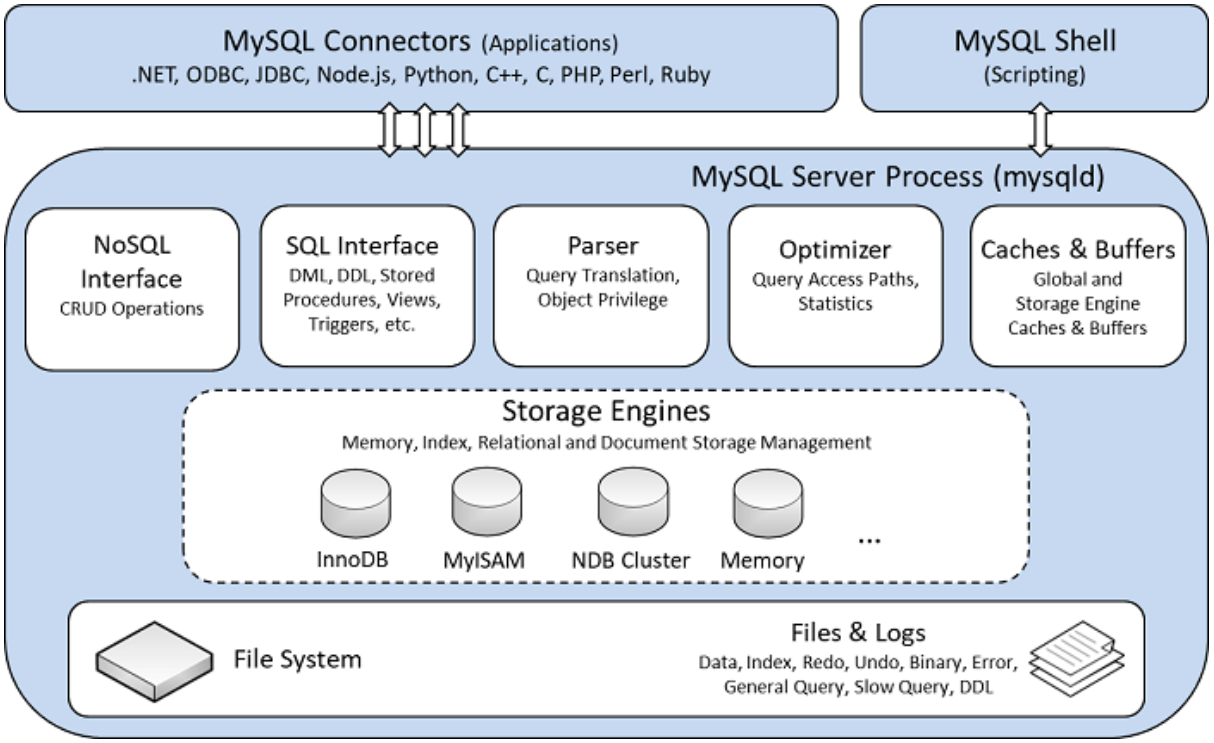
MySQL est un système de gestion de bases de données relationnelles distribué sous une double licence GPL et propriétaire.

Fondé par Michael Widenius, David Axmark et Allan Larsson en 1995 (première version le 23 mai 1995), il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public que par des professionnels.

Mysql est ensuite rachetée en janvier 2008 par Sun Microsystems qui sera ensuite acquis par Oracle Corporation, mettant entre les mains d'une même société les deux produits concurrents que sont Oracle Database et MySQL.

2. Architecture fonctionnelles du SGBD choisi

Dessins d'architecture



Les caches mémoires et leurs rôles

MySQL utilise plusieurs façons de mettre en cache des informations dans des tampons en mémoire afin d'augmenter les performances :

- InnoDB : maintient une zone de stockage appelée le pool de mémoire tampon pour la mise en cache des données et des index en mémoire. Le fonctionnement du pool tampon InnoDB, pour conserver en mémoire les données fréquemment consultées est un aspect important de l'optimisation de MySQL.
- Le cache des clés MyISAM : minimise les entrées/sorties sur disque, le moteur de stockage MyISAM exploite une stratégie utilisée par de nombreux systèmes de gestion de bases de données. Il utilise un mécanisme de cache pour conserver en mémoire les blocs de table les plus fréquemment utilisés.
- Mise en cache des instructions préparées et des programmes stockés : pour certaines instructions que l'on veut exécuter plusieurs fois au cours d'une session, le serveur convertit l'instruction en une structure interne et met cette structure en cache pour qu'elle soit utilisée pendant l'exécution. La mise en cache permet au serveur d'être plus efficace car elle évite les frais généraux de reconversion de l'instruction si elle est à nouveau nécessaire au cours de la session.

https://dev.mysql.com/doc/refman/8.0/en/buffering-caching.html

Les processus gravitant autour du cache mémoire et leur rôle

Dans MySQL, il y a 2 principaux processus qui gravitent autours du cache mémoire :

- Le moteur de stockage mémoire (MEMORY Storage Engine)

Le moteur de stockage mémoire (anciennement connu sous le nom de HEAP) créer des tables à usage spécial dont le contenu est stocké en mémoire. Comme les données sont vulnérables aux pannes, aux problèmes matériels ou aux coupures de courant, n'utilisez ces tables que comme zones de travail temporaires ou comme caches en lecture seule pour les données extraites d'autres tables.

Il est souvent utilisé pour des opérations impliquant des données transitoires et non critiques telles que la gestion des sessions ou la mise en cache. Il y a un stockage en mémoire pour un accès rapide et une faible latence.

Lorsque le serveur MySQL s'arrête ou redémarre, les données des tables MEMORY sont perdues.

- Le cache de requêtes MySQL

Le cache de requêtes stocke le texte d'une instruction ainsi que le résultat.
Si une instruction identique est reçue ultérieurement, le serveur récupère les résultats dans le cache de requête plutôt que d'analyser et d'exécuter à nouveau l'instruction.

Le cache de requêtes est partagé entre les sessions, de sorte qu'un ensemble de résultats généré par un client peut être envoyé en réponse à la même requête émise par un autre client.

https://dev.mysql.com/doc/refman/8.0/en/memory-storage-engine.html
https://dev.mysql.com/doc/refman/5.7/en/query-cache.html

Gestion des connexions

MySQL permet la création de comptes qui autorisent les utilisateurs clients à se connecter au serveur et à accéder aux données gérées par le serveur. La fonction principale 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.

Chaque compte peut se voir attribuer des informations d'authentification telles qu'un mot de passe pour contrôler les connexions. L'interface utilisateur 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. En interne, le serveur stocke les informations relatives aux privilèges dans les tables d'autorisation de la base de données système mysql. Le serveur MySQL lit le contenu de ces tables en mémoire au démarrage et fonde les décisions de contrôle d'accès sur les copies en mémoire des tables d'autorisation.

Le contrôle d'accès MySQL comporte deux étapes lorsque vous exécutez un programme client qui se connecte au serveur :

- Le serveur accepte ou rejette la connexion en fonction de votre identité et en vérifiant votre identité en fournissant le mot de passe correct.
- En supposant que vous puissiez vous connecter, le serveur vérifie chaque instruction que vous émettez pour déterminer si vous disposez des privilèges suffisants pour l'exécuter.

https://dev.mysql.com/doc/refman/8.0/en/access-control.html

Processus d’exécution des requêtes

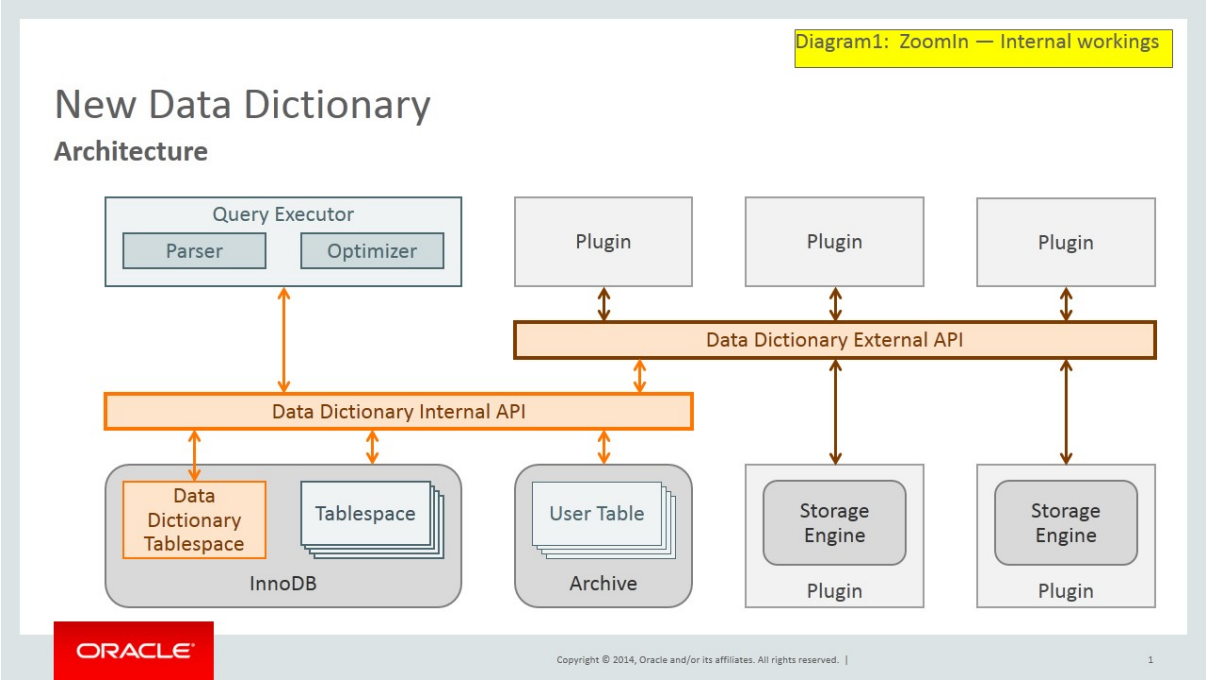
Pour le processus d’exécution des requêtes MySQL va utiliser le pré-filtrage. Celui-ci va limiter les informations d'événement collectées et est indépendant de tout utilisateur particulier.

Cependant, via les requêtes contenant des clause WHERE appropriées restreignant les informations d'événement à sélectionner parmi l'ensemble des événement disponibles après le pré-filtrage les utilisateurs individuels vont effectuer le post filtrage.

https://dev.mysql.com/doc/refman/8.0/en/performance-schema-queries.html

3. Le dictionnaire de données du moteur choisi

Depuis longtemps, il y a des plaintes concernant les déficiences du dictionnaire de données de MySQL.
Un nouveau dictionnaire de données amélioré pour MySQL est alors mis en place :



Les tables du dictionnaire de données sont protégées et ne sont accessibles que dans les builds de débogage de MySQL. Toutefois, MySQL prend en charge l'accès aux données stockées dans les tables du dictionnaire de données par le biais des tables INFORMATION_SCHEMA et des instructions SHOW.

https://mysqlserverteam.com/a-preview-on-lab-release-with-new-data-dictionary-in-mysql/
https://dev.mysql.com/doc/refman/8.0/en/data-dictionary.html

4. Organisation physique du SGBD

Il existe plusieurs extensions de fichier afin d'aider au fonctionnement du SGBD. Voici une liste non-exhaustive que l'on peut trouver, des types de fichiers présents dans l'infrastructure MySQL :

.FRM : Contient les informations de formatage ou les données structurelles d'une base de données MySQL; spécifie les tables stockées dans la base de données et définit les champs et la structure de chaque table. Les fichiers FRM sont sauvegardés avec un fichier .MYD, qui contient les données stockées dans la base de données…

.MYD : Fichier base de données créé avec MySQL, un système de gestion de base de données libre (open source) et très populaire; contient les données stockées à l'intérieur des lignes de la base de données. Les fichiers MYD sont sauvegardés avec un fichier .FRM correspondant qui contient les données de format de la tab...

.MYI : Index d'une table MyISAM créée dans une base de donnée MySQL; définit la structure de la table et inclut un compteur dans l'entête qui montre si la table a été fermée correctement ou pas. MyISAM est le moteur de stockage par défaut utilisé par MySQL; il gère les tables non transactionnelles et fournit des st...

.SQL : Fichier de base de données écrit en SQL (Structured Query Language en anglais ou Langage Structuré de Requêtes); peut être lu par n'importe quel programme de base de données compatible avec SQL, tel que FileMaker, MS Access, ou MySQL (communément utilisé sur des serveurs Web basé sur Linux)....

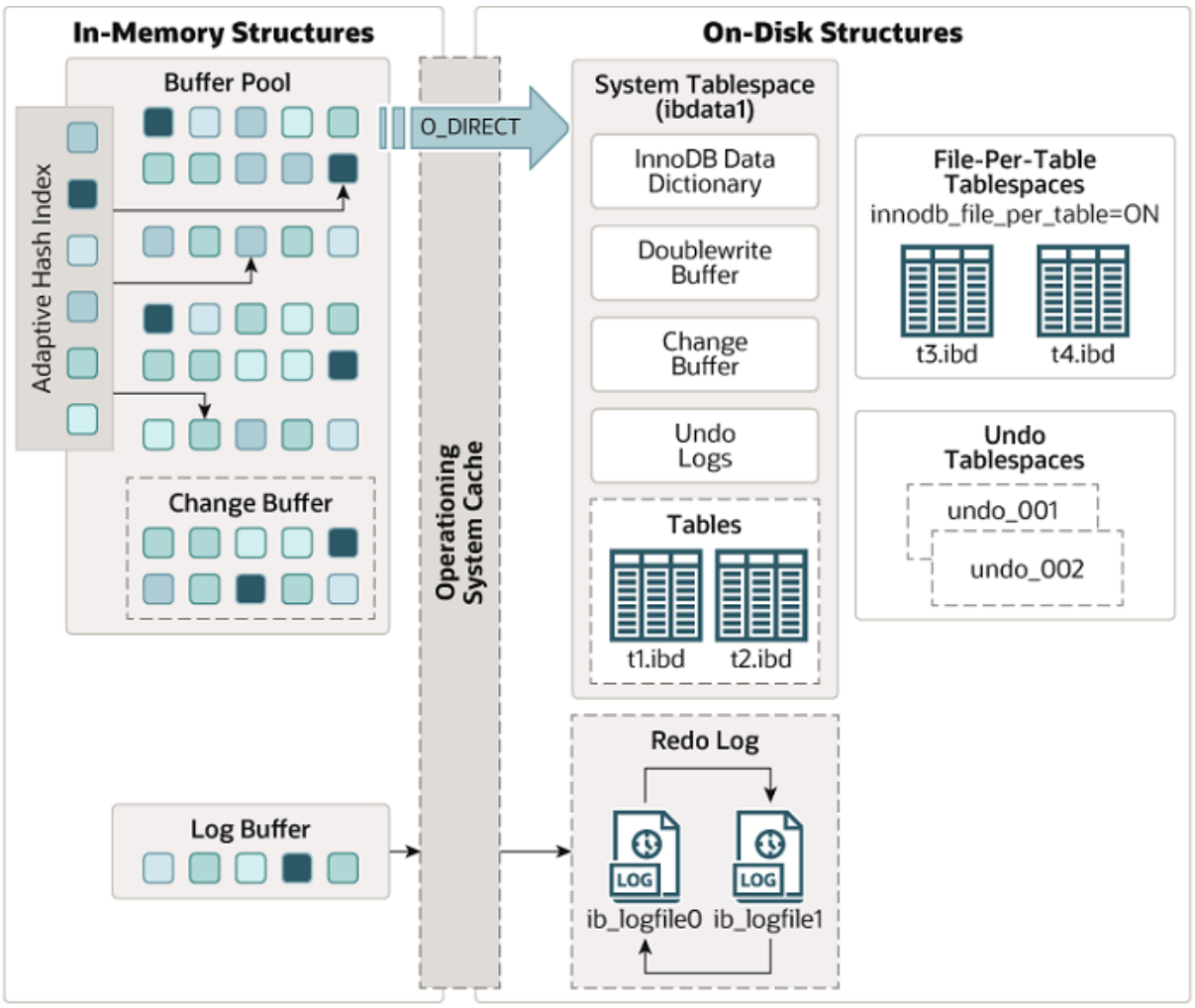
.IBD: Ces fichiers se trouvent généralement dans le répertoire de données.
Une table définie par l'utilisateur et ses données d'index correspondantes, dans InnoDB, sont stockées dans des fichiers qui ont une extension .ibd.

.SOCK : Toutes les connexions à la base de données MySQL sont gérées par un fichier spécial appelé fichier socket. Ce fichier facilite la communication entre les différents processus.

.PID : L'id de processus du serveur MySQL est écrit dans ce fichier.

.DB : Ce sont les fichiers avec les extensions .db stockent les données et les index du moteur de stockage BerkeleyDB.

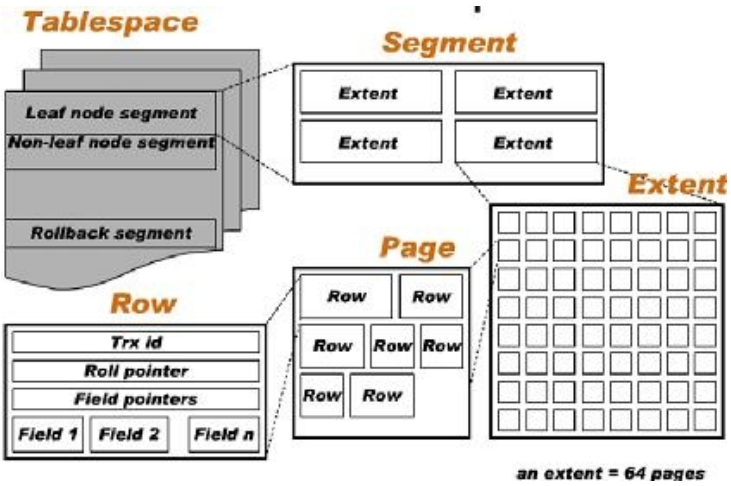
.LOG : il s'agit des erreurs systèmes, des erreurs des requêtes, des erreurs binaires qui sont relatives à la création de données ou de table orchestré par MySQL et les "slow query" qui sont présentes pour améliorer les performances.



https://www.geeksforgeeks.org/mysql-database-files/
http://man.hubwiz.com/docset/MySQL.docset/Contents/Resources/Documents/innodb-storage-engine.html
https://dev.mysql.com/doc/refman/5.6/en/innodb-architecture.html

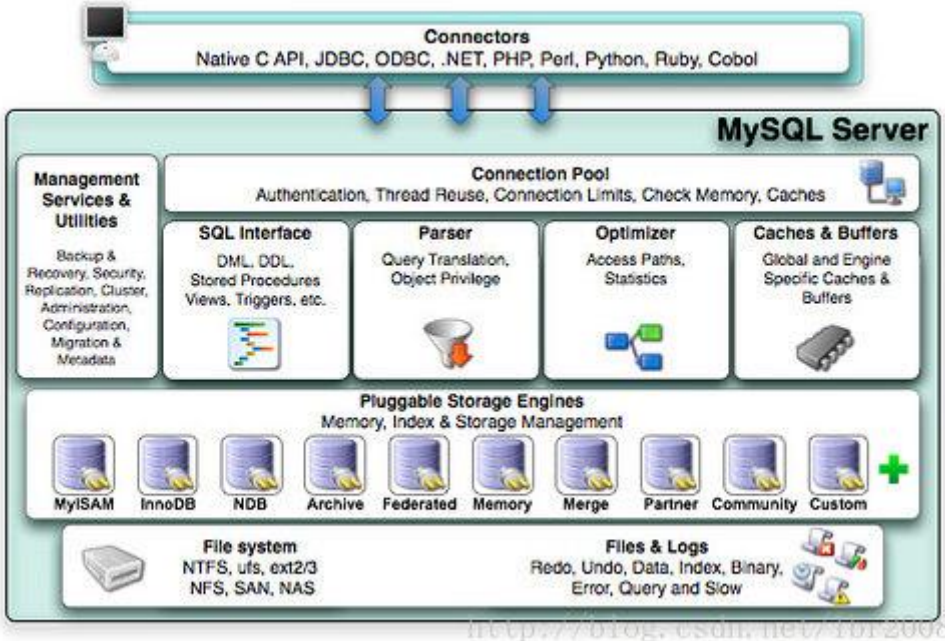
5. Organisation logique des données

Comment sont organisées de manière logique (tablespace / segments / block chez Oracle par exemple) les données des tables, les données des index, les données d'annulation et les données temporaires :



Dans le moteur de stockage InnodB, toutes les données sont stockées logiquement dans le tablespace. Il s'agit de l'unité logique de stockage la plus élevée du moteur. Quand Mysql stocke des tables, il sépare les informations comme les définitions de table et les index de données. Les tables sont stockées dans des fichiers .frm et les indexes dans des fichiers .idb.

Mysql a également ce qu'on peut appeler des données d'annulation, c'est-à-dire des données qu'il stock dans le cas où il y aurait une suppression ou une modification pour pouvoir revenir en arrière.
Enfin, mysql peut créer une table spéciale appelée Table temporaire qui nous permet de conserver des données temporaires. 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.



6. Gestion de la concurrence d’accès

Notion de transaction

MySQL (ici nous maintenons la version 5.6) 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. Voici la syntaxe de START TRANSACTION, COMMIT, et ROLLBACK :

Support des propriétés ACID

Dans MySQL, le moteur de stockage InnoDB prend en charge les fonctionnalités conformes à la norme ACID. Les sections suivantes traitent de la manière dont les fonctionnalités de MySQL, en particulier le moteur de stockage InnoDB, interagissent avec les catégories du modèle ACID :

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

- Le paramètre Autocommit.
- Instruction COMMIT.
- L'instruction ROLLBACK.
- Données opérationnelles des tables INFORMATION_SCHEMA.

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

- Le tampon de double écriture InnoDB.
- Récupération de crash InnoDB.

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

- Le paramètre Autocommit.
- L'instruction SET ISOLATION LEVEL.

- Les détails de bas niveau du verrouillage InnoDB. Pendant l'optimisation des performances, vous voyez ces détails par le biais des tables INFORMATION_SCHEMA.

Durabilité : L'aspect durabilité du modèle ACID implique que les fonctionnalités du logiciel MySQL interagissent avec votre configuration matérielle particulière. En raison des nombreuses possibilités qui dépendent des capacités de votre CPU, de votre réseau et de vos périphériques de stockage, cet aspect est le plus compliqué à gérer. Les fonctionnalités MySQL connexes sont les suivantes :

- Le tampon de double écriture InnoDB activé et désactivé par l'option de configuration innodb_doublewrite.
- Option de configuration innodb_flush_log_at_trx_commit.
- Option de configuration sync_binlog.
- Option de configuration innodb_file_per_table.
- Tampon d'écriture dans un périphérique de stockage, tel qu'un disque dur, un SSD ou une matrice RAID.
- Cache alimenté par batterie dans un périphérique de stockage.
- Le système d'exploitation utilisé pour exécuter MySQL, en particulier, son support de l'appel système fsync().
- Alimentation sans coupure (UPS) protégeant l'alimentation électrique de tous les serveurs informatiques et périphériques de stockage qui exécutent les serveurs MySQL et stockent les données MySQL.
- Votre stratégie de sauvegarde, 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 centre

https://www.w3resource.com/mysql/mysql-transaction.php#MSQLT

Technique de gestion de la concurrence d’accès

Il y a un accès concurrent lorsque plusieurs utilisateurs (transactions) accèdent en même temps à la même donnée dans une base de données.

Il faut alors s'assurer que l'exécution simultanée des transactions produit le même résultat que leur exécution séquentielle.

Le moteur de stockage MyISAM supporte les insertions simultanées afin de réduire la contention entre les lecteurs et les écrivains pour une table donnée : Si une table MyISAM ne présente pas de trous dans le fichier de données (lignes supprimées au milieu), une instruction INSERT peut être exécutée pour ajouter des lignes à la fin de la table en même temps que les instructions SELECT lisent des lignes de la table. S'il y a plusieurs instructions INSERT, elles sont mises en file d'attente et exécutées en séquence, en même temps que les instructions SELECT. Les résultats d'un INSERT concurrent peuvent ne pas être visibles immédiatement.

Pour pallier également au problème, on peut mettre en place des verrous pour restreindre ou interdire l'accès à certains éléments. Ce type de verrouillage est interne car il est entièrement réalisé par le serveur et n'implique aucun autre programme :

- MySQL utilise le verrouillage au niveau des lignes pour les tables InnoDB afin de prendre en charge l'accès en écriture simultanée par plusieurs sessions. Les verrous nécessaires se font au début de la transaction en émettant une instruction SELECT ... FOR UPDATE pour chaque groupe de lignes devant être modifiées, même si les instructions de modification des données interviennent plus tard dans la transaction.
- MySQL utilise le verrouillage au niveau des tables pour les tables MyISAM, MEMORY et MERGE, ce qui permet à une seule session de mettre à jour ces tables à la fois. Ce niveau de verrouillage rend ces moteurs de stockage plus adaptés aux applications en lecture seule, en lecture partielle ou à un seul utilisateur.
  Ces moteurs de stockage évitent les blocages en demandant toujours tous les verrous nécessaires en une seule fois au début d'une requête et en verrouillant toujours les tables dans le même ordre.

Les verrous permettent d'éviter les lignes fantômes, les lectures incohérentes, ...

https://dev.mysql.com/doc/refman/5.6/en/concurrent-inserts.html
https://dev.mysql.com/doc/refman/5.6/en/internal-locking.html

La 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 : ce sont des informations sérialisées du dictionnaire (SDI). InnoDB stocke ces données dans ses fichiers tablespace. NDBCLUSTER stocke les données SDI dans le dictionnaire NDB. D'autres moteurs de stockage stockent les données SDI dans des fichiers .sdi qui sont créés pour une table donnée dans le répertoire de la base de données de la table. Les données SDI sont générées dans un format JSON compact.

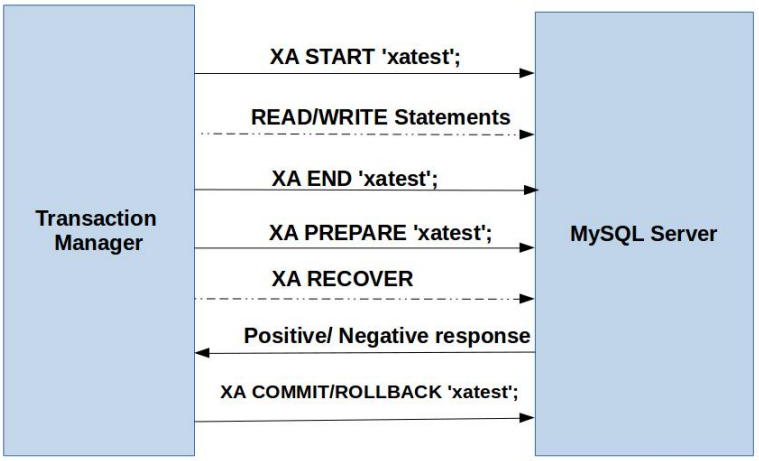
Les informations de dictionnaire sérialisées (SDI) sont présentes dans tous les fichiers de tablespace InnoDB, à l'exception des fichiers de tablespace temporaires et de tablespace d'annulation. Les enregistrements SDI d'un fichier tablespace InnoDB décrivent uniquement les objets de la table et du tablespace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

https://dev.mysql.com/doc/refman/8.0/en/serialized-dictionary-information.html

7. Gestion des transactions distribuées

Architecture

Les transactions XA sont des méthodes standardisées pour l'exécution de transactions via plusieurs ressources. Ces ressources peuvent être des bases de données ou tout autre système transactionnel. Le serveur MySQL supporte les requêtes SQL XA qui permettent aux utilisateurs de mener à bien une transaction distribuée SQL qui touchent plusieurs serveurs de base de données ou tout autre système supportant les requêtes SQL. Dans un tel cas, il en est de la responsabilité de l'utilisateur de coordonner les serveurs participants.
XA utilise un protocole de validation en deux phases, la première phase étant la demande de validation suivie de la validation proprement dite.
Le diagramme suivant illustre une transaction XA impliquant un gestionnaire de ressources :



http://delhay.iutlan.univ-rennes1.fr/doc/PHP/mysqlnd-ms.quickstart.xa_transactions.html
https://mysqlserverteam.com/improvements-to-xa-support-in-mysql-5-7/

Méthode pour garantir l'atomicité, la cohérence et la durabilité

XA (eXtended Architecture) est une norme créée par The Open Group pour le traitement des transactions distribuées. XA aborde le problème de la préservation des propriétés ACID au sein d'une transaction unique sur un ensemble distribué de ressources.

Du côté du client, il n'y a pas d'exigences particulières. L'interface XA vers un serveur MySQL consiste en des instructions SQL qui commencent par le mot-clé XA. Les programmes clients MySQL doivent être capables d'envoyer des instructions SQL et de comprendre la sémantique de l'interface des instructions XA. Il n'est pas nécessaire qu'ils soient liés à une bibliothèque client récente. Les bibliothèques clientes plus anciennes fonctionnent également.

Une transaction globale implique donc plusieurs actions qui doivent toutes se terminer avec succès en tant que groupe, ou être annulées en tant que groupe. 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.

https://mysqlserverteam.com/improvements-to-xa-support-in-mysql-5-7/
https://dev.mysql.com/doc/refman/8.0/en/xa.html

Traitement des pannes

Pour le traitement des pannes, nous analysons les actions mises en place pour éviter la réalisation de requêtes incomplètes.
Le système utilisé pour éviter ce genre de désagrément est un système de transaction.

Comme vu précédemment, les transactions sont une fonctionnalité permettant de sécuriser une application. Sans transactions, certaines opérations risqueraient d'être à moitié réalisées. Les transactions permettent de regrouper des requêtes dans des blocs, et de faire en sorte que tout le bloc soit exécuté en une seule fois, cela afin de préserver l'intégrité des données de la base.

Grâce au système de transaction, dans le cas d'une panne il ne peut pas y avoir de requêtes à moitié exécutées et donc une gestion des pannes en cours assez efficace.

Depuis la version 5.0 de MySQL, XA est supporté, il permet de réaliser des transactions dans Mysql avec un protocole de validation en deux phases, la première est une demande de validation et la seconde une validation proprement dite.
Une fois que les branches individuelles de la transaction globale ont terminé leur exécution, le protocole de validation entre en jeu.

https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql/1970063-transactions
https://mysqlserverteam.com/improvements-to-xa-support-in-mysql-5-7/

8. Gestion de la reprise sur panne

Les 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.

```
-- on désactive l'autocommit
SET AUTOCOMMIT =0;
-- on lance la transaction
START TRANSACTION;
-- on effectue cette simple requête sur notre table compte
UPDATE compte SET montant = montant + 20000 WHERE nom = 'vendeur';
-- cette requête retourne une erreur
UPDATE compte SET montant = montant - 20000 WHERE nom = 'machin';
-- on annule donc la transaction
ROLLBACK ;
```

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.

https://dev.mysql.com/doc/refman/8.0/en/innodb-autocommit-commit-rollback.html
https://dev.mysql.com/doc/refman/8.0/en/commit.html

La journalisation

Le serveur MySQL possède plusieurs journaux qui peuvent vous aider à découvrir l'activité en cours. Cela consiste à demander d'enregistrer dans un fichier toutes les requêtes de modification des données dans une base de données. Ainsi il est possible de recharger un journal.

Les différents types de journaux :

Journal des erreurs	Problèmes rencontrés lors du démarrage, de l'exécution ou de l'arrêt de mysqld
Journal général des requêtes	Connexions client établies et déclarations reçues des clients
Journal binaire	Déclarations qui modifient les données (également utilisé pour la réplication)
Journal des relais	Modifications de données reçues d'un serveur source de réplication
Journal des requêtes lentes	Requêtes dont l'exécution a pris plus de secondes que le temps de requête long_query_time.
Journal DDL (journal des métadonnées)	Opérations sur les métadonnées effectuées par les instructions DDL.

Par défaut, aucun journal n'est activé, le serveur écrit les fichiers de tous les journaux activés dans le répertoire de données. Vous pouvez forcer le serveur à fermer et à rouvrir les fichiers journaux.

https://www.editions-eni.fr/open/mediabook.aspx?idR=3c042dcc684026894b84aa71dfdd88a2
https://sites.google.com/site/gremiportfolio/patrimoine-informatique/journalisation-de-mysql

Les sauvegardes

MySQL a une variété de stratégies de sauvegarde comme :
Les types de sauvegardes :

- Logique (Les sauvegardes logiques sauvegardent les informations représentées sous forme de structure logique de base de données et de contenu. Ce type de sauvegarde est adapté aux petites quantités de données).
- Physique (Les sauvegardes physiques consistent en des copies brutes des répertoires et des fichiers qui stockent le contenu des bases de données. Ce type de sauvegarde convient aux bases de données importantes).

- Les sauvegardes à chaud ont lieu lorsque le serveur MySQL est en cours d'exécution, de sorte que les informations de la base de données peuvent être obtenues à partir du serveur (le serveur reste en fonctionnement mais où il est verrouillé contre toute modification des données alors que vous accédez aux fichiers de la base de données en externe)
- Les sauvegardes à froid ont lieu lorsque le serveur est arrêté.Incrémentielle, etc.

Méthodes de création de sauvegardes :

- Sauvegarde à chaud avec MySQL Enterprise Backup
- Réalisation de sauvegardes avec mysqldump (peut effectuer des sauvegardes et 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
- Faire des sauvegardes en copiant des fichiers de tables
- Sauvegarde binaire (en copiant tous les fichiers de table, tant que le serveur ne met rien à jour)

Mysql peut également planifier des sauvegardes, crypter des données, faire de la maintenance des tables, pour permettre la récupération de tables corrompues.

https://dev.mysql.com/doc/mysql-backup-excerpt/5.7/en/backup-and-recovery.html

Gestion de la reprise à chaud

La commande mysqlbackup, qui fait partie du composant MySQL Enterprise Backup, et permet de sauvegarder une instance MySQL en cours d'exécution, en perturbant le moins possible les opérations et en restant cohérent. Quand 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.

Gestion de la reprise à froid

La sauvegarde à froid consiste à copier les fichiers pendant l'arrêt du serveur MySQL.
On peut alors faire une sauvegarde physique qui consiste en tous les fichiers utilisés par InnoDB pour gérer ses tables. Utilisez la procédure suivante :

- Arrêt lent du serveur MySQL et vérifier qu'il s'arrête sans problèmes.
- Faire une copie de tous les fichiers de données InnoDB, de tous les fichiers .frm des tables InnoDB, tous les fichiers journaux InnoDB et les fichiers de configuration my.cnf dans un endroit sûr.

https://dev.mysql.com/doc/mysql-backup-excerpt/5.7/en/innodb-backup.html

9. Techniques d'indexation

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 et si la table est très grande, cela coûte cher.
S'il y a un index pour les colonnes, MySQL peut rapidement déterminer la position à rechercher au milieu du fichier de données sans regarder toutes les données, ce qui est plus rapide que de tout lire.

La plupart des index MySQL sont stockés dans des B-trees. Il existe également d'autres cas même s'ils sont plus rares : les arbres R, les tables MEMORY avec les index de hachage ou des listes inversées pour les index FULLTEXT utilisés par InnoDB.

Un index B-tree peut être utilisé pour comparer des colonnes dans les expressions qui utilisent les opérateurs =, >, >=, <, <= , BETWEEN ou LIKE si l'argument de LIKE est une chaîne constante qui ne commence pas par un caractère générique.
Exemple :

```
SELECT * FROM tbl_name WHERE key_col LIKE 'Patrick%';
SELECT * FROM tbl_name WHERE key_col LIKE 'Pat%_ck%';
```

Pour les autres cas plus rares, nous avons donc :
Les index de hachage sont utilisés uniquement pour les comparaisons d'égalité qui utilisent les opérateurs = ou <=>. Ils ne sont pas utilisés pour les opérateurs de comparaison tels que < qui trouvent une plage de valeurs. Les systèmes qui s'appuient sur ce type de recherche à valeur unique sont connus sous le nom de "key-value stores". Pour utiliser MySQL pour de telles applications, utilisez les index de hachage autant que possible.

Pour un R-tree, la recherche et la traversée de l'arbre R sont différentes de celles de l'arbre B dans le sens où un critère de recherche peut être satisfait dans plusieurs feuilles de différents chemins de recherche.

https://dev.mysql.com/doc/refman/8.0/en/mysql-indexes.html

10. Optimisation de requêtes

Les indexes sont utilisés pour trouver des lignes de résultat avec une valeur spécifique, très rapidement. Sans indexes, MySQL doit lire successivement toutes les lignes et à chaque fois, faire les comparaisons nécessaires pour extraire un résultat pertinent. Plus la table est grosse, plus c'est coûteux en ressources. Si la table dispose d'un index pour les colonnes utilisées, MySQL peut alors trouver rapidement les positions des lignes dans le fichier de données sans avoir à parcourir toute la table.

La commande EXPLAIN permet d'auditer une requête SQL et ainsi connaître le nombre de lignes (rows) parcourues pour exécuter la requête. Si ce nombre est très élevé, l'utilisation d'un index peut s'avérer utile afin d'optimiser la requête. L'ajout d'index se fait la plupart du temps sur les colonnes utilisées dans la clause WHERE d'une requête SQL.

Pour optimiser les requêtes on peut également :

- Eviter si possible les SELECT * et réduisez le nombre de champs, afin de réduire les données chargées en mémoire.
- Remplacer vos clauses WHERE … IN par WHERE EXISTS.
- Compter les requêtes sur chaque page, un grand nombre de requêtes peut signifier un « problème N+1 », c'est à dire une requête SELECT placée dans une boucle
- Utiliser les requêtes préparées ou les procédures stockées facilite la mise en cache des requêtes en interne par MySql et vous assure un bon niveau de sécurité.
- Utiliser la clause EXPLAIN pour comprendre le fonctionnement d'une requête et quelles sont les clauses qui impactent ses performances.

Enfin, il y a l'optimiseur de requête qui est directement intégré à MYSQL. Cette optimiseur va permettre par exemple d'augmenter la rapidité des requêtes déjà effectuées. Il est géré selon différents plans qui vont s'appliquer à chaque requête.

https://www.infomaniak.com/fr/support/faq/377/mysql-optimiser-les-requetes#:~:text=Le%20nombre%20et%20la%20taille,de%20la%20base%20augmente%20consid%C3%A9rablement.
https://ircf.fr/actualites/optimiser-votre-base-de-donnees-mysql/