DOSSIER DE RECHERCHE SGBD

ORACLE

MySQL

ANNÉE UNIVERSITAIRE 2020-2021

UNIVERSITÉ CÔTE D'AZUR

miage
NICE
SOPHIA
ANTIPOLIS

Table des matières

PARTIE 1: EVALUATION D'UN SGBD RELATIONNEL DU MARCHE - MYSQL

1. Identification du SGBD

- Nom : MySQL
- Date de parution : 1994
- Créateur : Michael Widenius
- Propriétaire : MySQL AB a été acheté en 2008 par *Sun Microsystems*. En 2009, *Sun Microsystems* a été acquis par *Oracle Corporation*.
- Modèles de données supportées : modèle de données relationnel

2. Architecture fonctionnelle du SGBD

2.1. Dessin d'architecture

MySQL peut alimenter des applications embarquées, des entrepôts de données, des logiciels d'indexation et de livraison de contenu, des systèmes redondants hautement disponibles, le traitement des transactions en ligne (OLTP)...

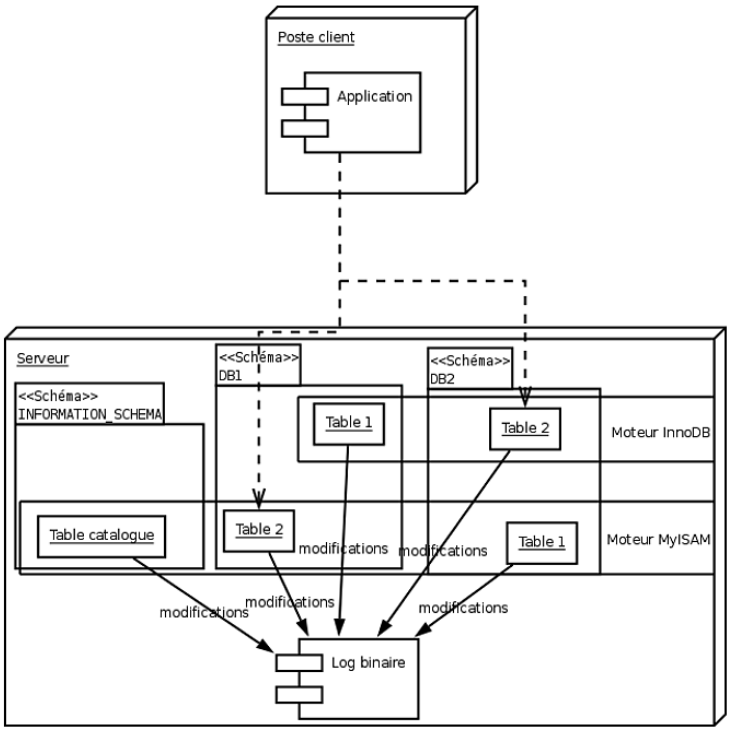
La couche supérieure contient les services qui ne sont pas uniques à MySQL. Ce sont des services dont la plupart des outils ou serveurs client / serveur basés sur le réseau ont besoin: gestion des connexions, authentification, sécurité, etc.

La deuxième couche est celle où les choses deviennent intéressantes. Une grande partie des cerveaux de MySQL sont ici, y compris le code pour l'analyse des requêtes, l'analyse, l'optimisation, la mise en cache et toutes les fonctions intégrées (par exemple, les dates, les heures, les mathématiques et le cryptage). Toutes les fonctionnalités fournies par les moteurs de stockage vivent à ce niveau: procédures stockées, déclencheurs et vues, par exemple.

La troisième couche contient les moteurs de stockage. Ils sont responsables du stockage et de la récupération de toutes les données stockées « dans » MySQL. Comme les différents systèmes de fichiers disponibles pour GNU / Linux, chaque moteur de stockage a ses propres avantages et inconvénients. Le serveur communique avec eux via l'API du moteur de stockage. Cette interface masque les

différences entre les moteurs de stockage et les rend largement transparentes au niveau de la couche de requête. L'API contient quelques douzaines de fonctions de bas niveau qui exécutent des opérations telles que « commencer une transaction » ou « récupérer la ligne contenant cette clé primaire ».

Les moteurs de stockage n'analysent pas SQL et ne communiquent pas entre eux; ils répondent simplement aux demandes du serveur.



Les distributions MySQL incluent plusieurs composants qui implémentent des extensions serveur :

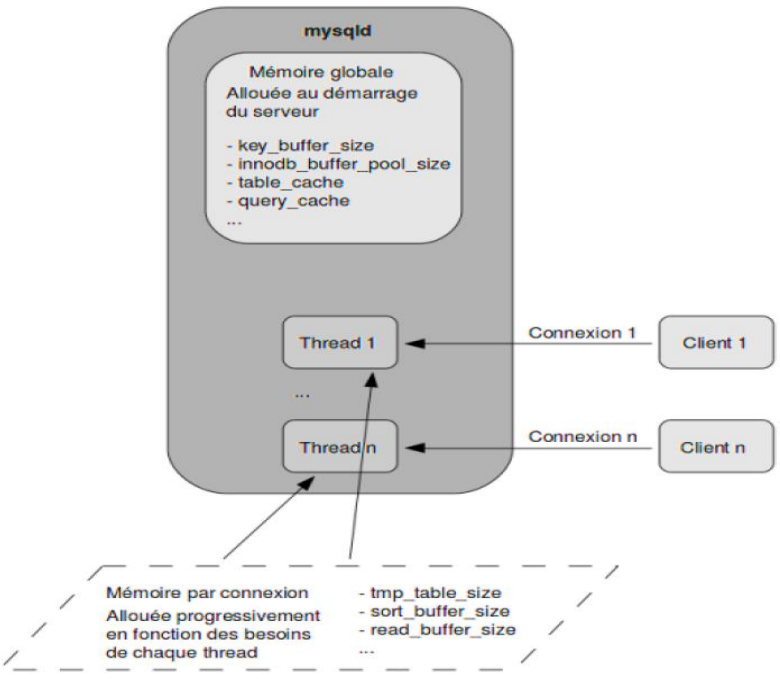
- Composants de configuration de la journalisation des erreurs.
- Un composant pour vérifier les mots de passe.
- Les composants du porte-clés fournissent un stockage sécurisé pour les informations sensibles
- Un composant qui permet aux applications d'ajouter leurs propres événements de message au journal d'audit.

- Un composant qui implémente une fonction définie par l'utilisateur pour accéder aux attributs de requête.

Ce sont les principaux composants de MySQL :

- Le serveur MySQL. C'est le serveur de base de données. Il traite toutes les requêtes et manipule les bases de données et les tables.
- Les clients MySQL. Ce sont des programmes qui communiquent avec le serveur.
- Répertoire de données. C'est l'endroit où MySQL stocke les bases de données.
- Moteur de stockage. Gère la manière dont les informations sont organisées, stockées et accessibles. MySQL utilise une architecture de moteur de stockage enfichable. Des exemples de moteurs de stockage sont MyISAM, InnoDB, Memory ou Csv.

2.2. Les caches mémoire et leurs rôles



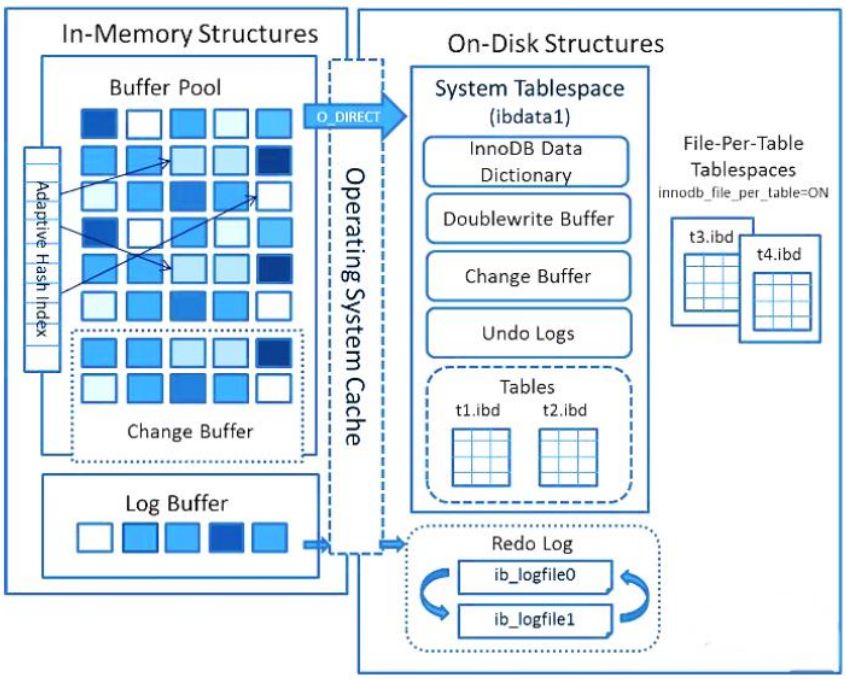
L'allocation mémoire du serveur peut être décomposée en deux catégories. La première est l'allocation par instance, qui est la mémoire allouée en une seule fois par le serveur au démarrage. Elle est partagée par le processus serveur mysqld et les threads, c'est-à-dire les connexions clientes. Elle comprend entre autres, la taille utilisée par le cache d'index des tables MyISAM (key_buffer_size), la taille du cache de données d'InnoDB (innodb_buffer_pool_size), la taille du cache de tables (table_cache), la taille du cache de requêtes (query_cache).

La deuxième catégorie est l'allocation par threads. C'est de la mémoire qui est allouée et désallouée dynamiquement en fonction des besoins de chaque client. Elle est composée notamment de la taille occupée par les données contenues dans les buffers utilisés par les clients tels que le sort_buffer_size ou le read_buffer_size. On y retrouve aussi la taille des tables temporaires (tmp_table_size), la taille maximale des tables Memory (max_heap_table_size), etc.

2.3. Les processus gravitant autour du cache mémoire et leur rôle

MySQL utilise le moteur de mémoire en interne lors du traitement des requêtes qui nécessitent une table temporaire pour contenir les résultats intermédiaires. Si le résultat intermédiaire devient trop grand pour une table mémoire, MySQL le convertira en table MyISAM sur le disque.

Structures en mémoire InnoDB et structures sur disque

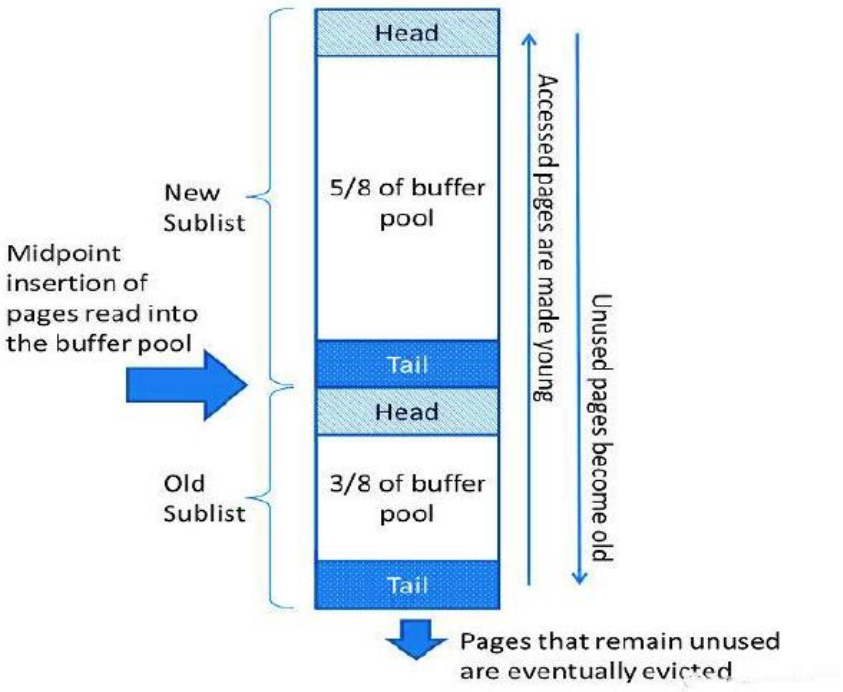


La zone mémoire SGA correspondant à Oracle dans MySQL est appelée pool de mémoire tampon InnoDB (innodb_buffer_pool) .

Le moteur de stockage InnoDB utilise le pool de tampons pour mettre en cache les données de table et les index en mémoire. Lors du traitement des données, on peut manipuler directement les données du pool de tampons pour améliorer la vitesse de traitement d'InnoDB. Les données du pool de mémoire tampon suivent généralement le format de page. Chaque page contient plusieurs lignes de données.

Le pool de mémoire tampon peut être considéré comme une liste liée de pages et l'algorithme LRU (dernier usage récent) est utilisé pour gérer la liste de données du pool de mémoire tampon.

Lorsqu'un nouvel espace est nécessaire pour ajouter de nouvelles pages au pool de mémoire tampon, les données les moins récemment utilisées seront éliminées.



En fait, la composition de la mémoire MySQL est similaire à celle d'Oracle, et elle peut également être divisée en SGA (System Global Area) et PGA (Program Cache Area).

Idéalement, il est conseillé de définir la taille du pool de mémoire tampon sur une valeur aussi grande que possible, laissant suffisamment de mémoire pour que les autres processus sur le serveur s'exécutent sans pagination excessive.

2.4 Gestion des connexions

Cette section décrit les aspects de la façon dont le serveur MySQL gère les connexions client.

2.4.1. Interfaces réseau et threads du gestionnaire de connexions

Le serveur est capable d'écouter les connexions client sur plusieurs interfaces réseau. Les threads du gestionnaire de connexion gèrent les demandes de connexion client sur les interfaces réseau que le serveur écoute :

- Sur toutes les plates-formes, un thread de gestionnaire gère les demandes de connexion TCP/IP.

- Sous Unix, le même thread de gestionnaire gère également les demandes de connexion de fichiers socket Unix.

- Sous Windows, un thread de gestionnaire gère les demandes de connexion à mémoire partagée et un autre gère les demandes de connexion à canal nommé.

- Sur toutes les plates-formes, une interface réseau supplémentaire peut être activée pour accepter les demandes de connexion TCP/IP administratives. Cette interface peut utiliser le thread gestionnaire qui gère les requêtes TCP/IP « ordinaires » , ou un thread séparé.

Le serveur ne crée pas de threads pour gérer les interfaces qu'il n'écoute pas. Par exemple, un serveur Windows sur lequel la prise en charge des connexions de canal nommé n'est pas activée ne crée pas de thread pour les gérer.

Les plugins ou composants de serveur individuels peuvent implémenter leur propre interface de connexion :

- X Plugin permet à MySQL Server de communiquer avec les clients à l'aide du protocole X. Voir Section 20.5, « Plugin X » .

2.4.2. Gestion des threads de connexion client

Les threads du gestionnaire de connexions associent chaque connexion cliente à un thread qui lui est dédié qui gère l'authentification et le traitement des demandes pour cette connexion. Les threads du gestionnaire créent un nouveau thread lorsque cela est nécessaire, mais essayez d'éviter de le faire en consultant d'abord le cache des threads pour voir s'il contient un thread qui peut être utilisé pour la connexion. Lorsqu'une connexion se termine, son thread est renvoyé au cache de threads si le cache n'est pas plein.

Dans ce modèle de thread de connexion, il y a autant de threads que de clients actuellement connectés, ce qui présente certains inconvénients lorsque la charge de travail du serveur doit évoluer pour gérer un grand nombre de connexions. Par exemple, la création et l'élimination des threads deviennent coûteuses. De plus, chaque thread nécessite des ressources serveur et noyau, telles que l'espace de pile. Pour accueillir un grand nombre de connexions simultanées, la taille de la pile par thread doit rester petite, ce qui conduit à une situation où elle est soit trop petite, soit le serveur consomme de grandes quantités de mémoire. L'épuisement d'autres ressources peut également se produire et les frais généraux de planification peuvent devenir importants.

MySQL Enterprise Edition inclut un plugin de pool de threads qui fournit un modèle alternatif de gestion des threads conçu pour réduire les frais généraux et améliorer les performances. Il implémente un pool de threads qui augmente les performances du serveur en gérant efficacement les threads d'exécution des instructions pour un grand nombre de connexions clientes.

Pour contrôler et surveiller la façon dont le serveur gère les threads qui gèrent les connexions client, plusieurs variables système et d'état sont pertinentes.

- La *thread_cache_size* variable système détermine la taille du cache de thread. Par défaut, le serveur dimensionne automatiquement la valeur au démarrage, mais elle peut être définie explicitement pour remplacer cette valeur par défaut. Une valeur de 0 désactive la mise en cache, ce qui entraîne la configuration d'un thread pour chaque nouvelle connexion et sa suppression lorsque la connexion se termine. Pour activer la N mise en cache des threads de connexion inactifs, il faut définir la *thread_cache_size* sur N au démarrage du serveur ou à l'exécution. Un thread de connexion devient inactif lorsque la connexion cliente à laquelle il était associé se termine.

- Pour surveiller le nombre de threads dans le cache et le nombre de threads créés car un thread n'a pas pu être extrait du cache, il faut vérifier les variables d'état *Threads_cached* et *Threads_created*.

- Lorsque la pile de threads est trop petite, cela limite la complexité des instructions SQL que le serveur peut gérer, la profondeur de récursivité des procédures stockées et d'autres actions consommatrices de mémoire. Pour définir une taille de pile d' N octets pour chaque thread, il faut démarrer le serveur avec *thread_stack* la valeur N.

2.4.3. Gestion du volume de connexion

Pour contrôler le nombre maximum de clients que le serveur autorise à se connecter simultanément, définissez la *max_connections* variable système au démarrage du serveur ou à l'exécution. Il peut être nécessaire d'augmenter max_connections si plusieurs clients tentent de se connecter simultanément, alors le serveur est configuré pour gérer. Si le serveur refuse une connexion car la *max_connections* limite est atteinte, il incrémente la *Connection_errors_max_connections* variable d'état.

mysqld autorise en fait *max_connections* + 1 connexions client. La connexion supplémentaire est réservée à l'usage des comptes qui ont le *CONNECTION_ADMIN privilège* (ou le *SUPER privilège obsolète*). En accordant le privilège aux administrateurs et non aux utilisateurs normaux (qui ne devraient pas en avoir besoin), un administrateur peut se connecter au serveur et l'utiliser *SHOW PROCESSLIST* pour diagnostiquer les problèmes même si le nombre maximum de clients non privilégiés est connecté.

Depuis MySQL 8.0.14, le serveur autorise également les connexions administratives sur une interface réseau administrative, qu'il faut configurer à l'aide d'une adresse IP et d'un port dédiés.

Le plug-in de réplication de groupe interagit avec MySQL Server à l'aide de sessions internes pour effectuer des opérations d'API SQL. Dans les versions de MySQL 8.0.18, ces sessions comptent dans la limite de connexions client spécifiée par la *max_connectionsvariable* système du serveur. Dans ces versions, si le serveur a atteint la *max_connections* limite lorsque la réplication de groupe est démarrée ou tente d'effectuer une opération, l'opération échoue et la réplication de groupe ou le serveur lui-même peut s'arrêter. Depuis MySQL 8.0.19, les sessions internes de *Group Replication* sont gérées séparément des connexions client, elles ne comptent donc pas dans la *max_connections* limite et ne sont pas refusées si le serveur a atteint cette limite.

Le nombre maximum de connexions client prises en charge par MySQL (c'est-à-dire la valeur maximale *max_connections* pouvant être définie) dépend de plusieurs facteurs :

- La qualité de la bibliothèque de threads sur une plate-forme donnée.
- La quantité de RAM disponible.
- La quantité de RAM est utilisée pour chaque connexion.
- La charge de travail de chaque connexion.
- Le temps de réponse souhaité.

- Le nombre de descripteurs de fichiers disponibles.

Linux ou Solaris devrait être capable de prendre en charge au moins 500 à 1 000 connexions simultanées de manière routinière et jusqu'à 10 000 connexions si on dispose de plusieurs gigaoctets de RAM et que la charge de travail de chacun est faible ou que le temps de réponse cible est peu exigeant.

L'augmentation de la *max_connections* valeur augmente le nombre de descripteurs de fichiers requis par mysqld . Si le nombre requis de descripteurs n'est pas disponible, le serveur réduit la valeur de max_connections.

L'augmentation de la *open_files_limit* variable système peut être nécessaire, ce qui peut également nécessiter d'augmenter la limite du système d'exploitation sur le nombre de descripteurs de fichiers pouvant être utilisés par MySQL.

2.5. Processus d'exécution des requêtes

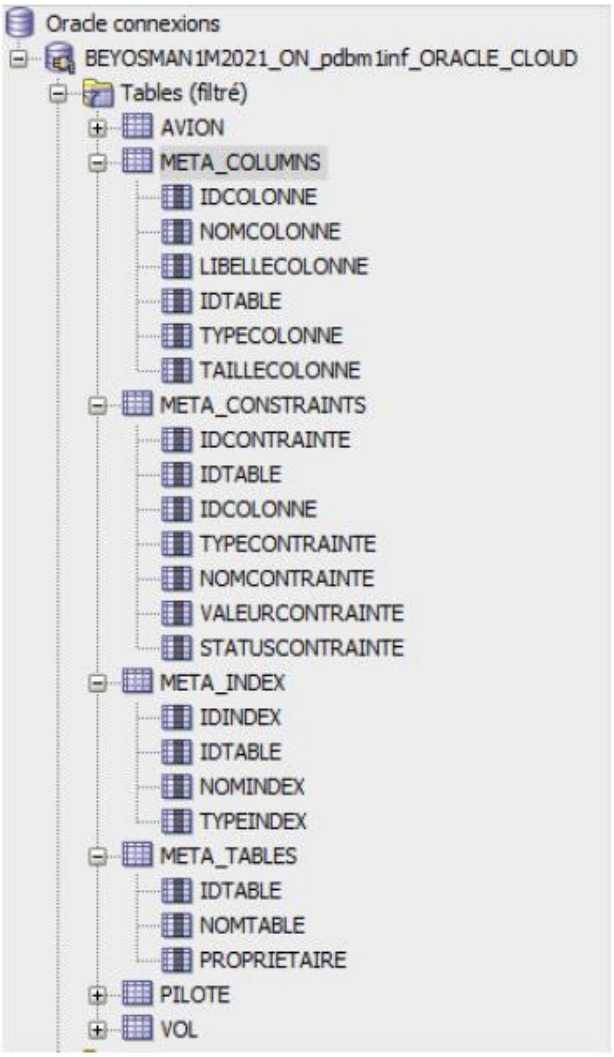
Le Langage de Définition de Données est utilisé pour créer et supprimer des objets dans la base de données (tables, contraintes d'intégrité, vues, etc.).

Le Langage de Manipulation de Données est utilisé pour la recherche, l'insertion, la mise à jour et la suppression de données. Le LMD est basé sur les opérateurs relationnels, auxquels sont ajoutés des fonctions de calcul d'agrégats et des instructions pour réaliser les opérations d'insertion, mise à jour et suppression.

Les requêtes sont processées par un compilateur fournissant des plans d'opérations à une machine d'exécution des requêtes.

3. Le dictionnaire de données

Voici les différentes tables systèmes qui permettent de mémoriser les différentes informations.



4. Organisation physique du SGBD

Les données sont stockées dans des fichiers, chaque table a trois fichiers (.frm, .MYD et .MYI).

- **Fichier Format BDD : extension .FRM** : il 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; les deux fichiers sont nécessaires pour que MySQL reconnaisse la base de données.

- **Fichier Données BD: extension .MYD :** c'est un fichier de base de données créé avec MySQL, un système de gestion de base de données libre (open source) et très populaire; il 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 table, sa structure, ainsi qu'un fichier .MYI qui sert d'index de la base de données.

- **Fichier Index MyISAM : extension .MYI :** c'est le fichier d'index d'une table MyISAM créée dans une base de données MySQL; il 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 stockages de données à grande vitesse ainsi que des opérations de récupération.

- **Données Langage Structuré de Requêtes : extension .SQL :** c'est un 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).

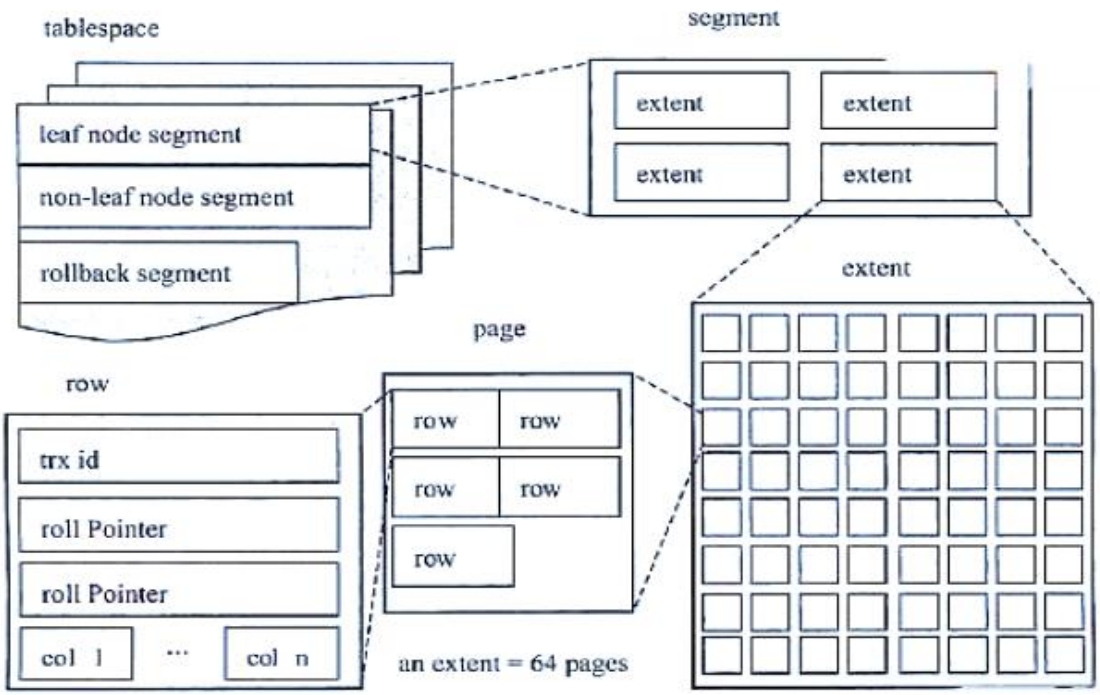
La base de données MySQL se compose des fichiers suivants :

- **my.cnf** : fichier de paramètres pour l'instance de la base de données. Il contient des informations sur l'emplacement des données, des archives, des fichiers redo et autres fichiers journaux. Il contient également des paramètres de mémoire pour l'instance et des informations sur l'hôte distant pour la réplication des données.

- **mysql-bin.index** : ce fichier contient des informations sur les journaux binaires (journaux d'archive dans oracle appelés journaux bin dans MySQL). Ces informations sont stockées dans le fichier de contrôle dans oracle.
- **Fichier Redo Log**
- **Fichiers journaux** : journalisation ou historique des événements ou log qui Inclut principalement le journal des erreurs, le journal binaire (ces fichiers contiennent toutes les transactions ou les déclarations de retour en arrière effectuées sur la base de données / le disque), le journal des requêtes, le journal des requêtes lentes, le journal de rétablissement innodb, le journal d'annulation innodb, etc.

Il existe également les fichiers socket. Le fichier socket n'est disponible que dans l'environnement Unix / Linux. L'utilisateur peut utiliser directement le socket Unix pour se connecter à MySQL au lieu du réseau TCP / IP dans l'environnement Unix / Linux.

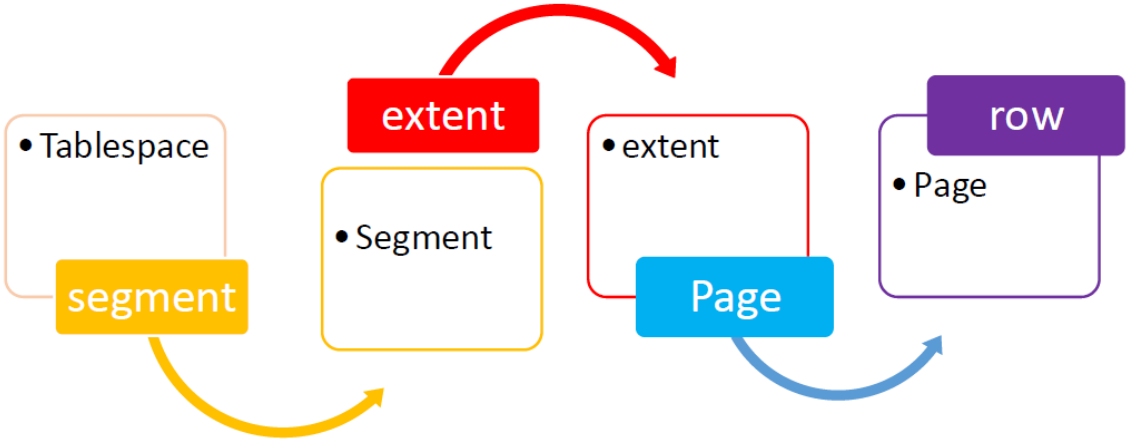
5. Organisation logique des données



Dans toute l'architecture de la base de données, nous pouvons utiliser différents moteurs de stockage pour stocker les données, et la plupart des moteurs de stockage

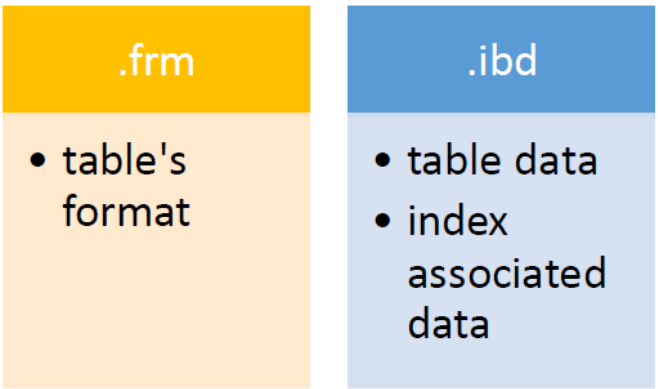
stockent les données sous forme binaire; cette section présentera comment les données sont stockées dans InnoDB (MySQL).

Dans le moteur de stockage InnoDB, toutes les données sont stockées logiquement dans l'espace table. L'espace table (tablespace) est l'unité logique de stockage la plus élevée du moteur de stockage. Sous l'espace table, il comprend des segments, des étendues, page.



Tous les tablespaces de la même instance de base de données ont la même taille de page ; par défaut, les tailles de page dans les tablespaces sont toutes de 16KB, bien que la taille par défaut puisse être modifiée en changeant l'option innodb_page_size, en notant que des tailles de page différentes résulteront éventuellement en des tailles de zone différentes aussi.

Lorsque MySQL utilise InnoDB pour stocker des tables, il stocke séparément des informations telles que les définitions de table et les index de données, les premières étant stockées dans des fichiers .frm et les seconds dans des fichiers .ibd.



Pour la base de données MySQL, les données sont stockées dans des fichiers, et afin de localiser rapidement un enregistrement dans une table pour une requête ou une modification, nous devons stocker ces données dans une certaine structure de données, que nous appelons index.

Un index, semblable à la table des matières d'un livre, vous permet de trouver immédiatement le contenu correspondant à partir d'un numéro de page dans la table des matières.

Avantages des index :

- Triés par nature.
- Recherche rapide.

Inconvénients d'un index :

- Prend de l'espace.
- Réduit la vitesse de mise à jour des tables

Dans InnoDB, il existe des index clustérisés et des index ordinaires

- L'index cluster est construit sur la base de la clé primaire et le nœud feuille stocke la ligne d'enregistrements correspondant à la clé primaire .
- L'index ordinaire est construit selon la colonne lorsque l'index est déclaré, et le nœud feuille stocke la valeur de la clé primaire correspondant à cette ligne d'enregistrements

Il existe deux cas particuliers d'un index unique et d'un index conjoint dans l'index ordinaire.

- L'index unique vérifiera si la valeur de la colonne correspondant à l'index existe déjà lors de l'insertion et de la modification

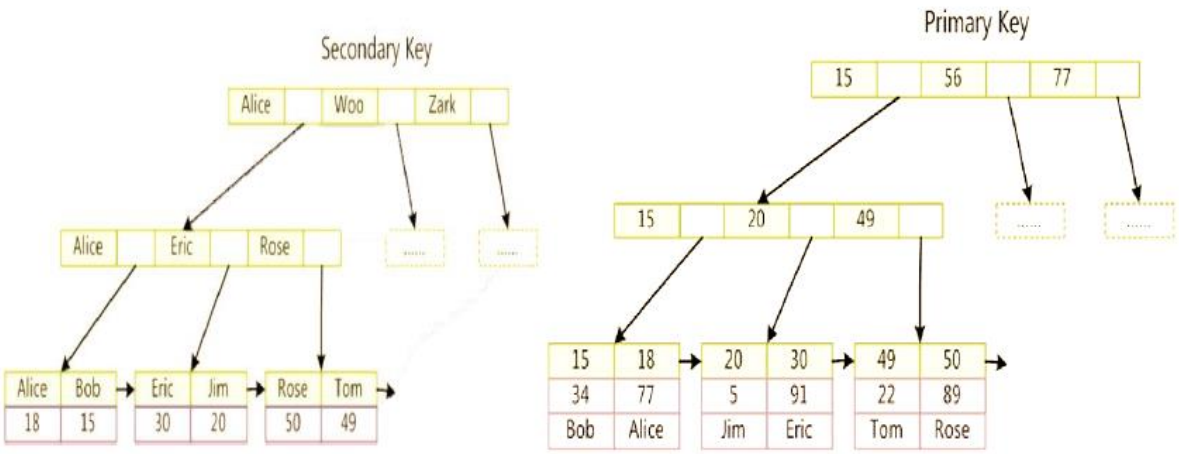
- L'index conjoint construira l'index après avoir concaténé les valeurs des deux colonnes dans l'ordre dans lequel elles ont été déclarées.

La structure de données d'index dans Innodb est un arbre B +, et les données sont organisées de manière ordonnée. Les données correspondantes sont trouvées couche par couche du nœud racine au nœud feuille .

Les données de chaque ligne de la table sont organisées et stockées dans un index clustérisé, c'est pourquoi on l'appelle une table organisée par index.

La valeur stockée dans l'index ordinaire est la valeur de la clé primaire. Si la clé primaire est une longue chaîne et que de nombreux index ordinaires sont créés, l'index ordinaire occupera beaucoup d'espace physique .

Exemple : Lors d'une interrogation avec Name = Alice, la valeur de clé primaire correspondante est d'abord 18, puis 18 est utilisée pour trouver le contenu de l'enregistrement de Name = Alice qui est 77 et Alice dans l'index cluster ci-dessous.



6. Gestion de la concurrence d'accès

Chaque fois que plusieurs requêtes doivent modifier des données en même temps, le problème du contrôle d'accès concurrentiel se pose. MySQL doit le faire à deux niveaux: le niveau du serveur et le niveau du moteur de stockage.

La solution à ce problème classique du contrôle d'accès concurrentiel est assez simple. Les systèmes qui gèrent un accès simultané en lecture / écriture implémentent généralement un système de verrouillage qui se compose de deux types de verrous.

Ces verrous sont généralement connus sous le nom de verrous partagés et verrous exclusifs ou verrous de lecture et verrous d'écriture. Les verrous de lecture sur une ressource sont partagés ou non bloquants: de nombreux clients peuvent lire à partir d'une ressource en même temps et ne pas interférer les uns avec les autres. Les verrous d'écriture, en revanche, sont exclusifs - c'est-à-dire qu'ils bloquent à la fois les verrous de lecture et les autres verrous d'écriture - car la seule politique sûre est d'avoir un seul client écrivant sur la ressource à un moment donné et d'empêcher toutes les lectures lorsqu'un client est en train d'écrire.

Dans le monde des bases de données, le verrouillage se produit tout le temps: MySQL doit empêcher un client de lire une donnée pendant qu'un autre la modifie. Il effectue cette gestion des verrous en interne de manière transparente la plupart du temps.

Une façon d'améliorer la concurrence d'une ressource partagée consiste à être plus sélectif sur ce qu'il y a à verrouiller. Plutôt que de verrouiller la totalité de la ressource, il est possible de verrouiller uniquement la partie contenant les données à modifier. La réduction de la quantité de données à verrouiller à un moment donné permet aux modifications d'une ressource donnée de se produire simultanément, tant qu'elles ne sont pas en conflit les unes avec les autres.

La stratégie de verrouillage la plus basique disponible dans MySQL, et celle avec la plus faible surcharge, est le verrouillage de table.

Le style de verrouillage qui offre la plus grande simultanéité (et qui entraîne le plus de frais généraux) est l'utilisation de verrous de ligne.

7. Gestion des transactions distribuées

Une transaction est un groupe de requêtes SQL traitées de manière atomique , comme une seule unité de travail. Si le moteur de base de données peut appliquer l'ensemble du groupe de requêtes à une base de données, il le fait, mais si l'une d'entre elles ne peut pas être effectuée en raison d'un plantage ou pour une autre raison, aucune d'elles n'est appliquée.

MySQL fonctionne en AUTOCOMMITmode par défaut.

MySQL ne gère pas les transactions au niveau du serveur. Au lieu de cela, les moteurs de stockage sous-jacents implémentent eux-mêmes les transactions. Cela signifie qu'il est impossible de mélanger de manière fiable différents moteurs en une seule transaction.

Voici des critères auxquels un système de traitement des transactions doit satisfaire:

- Atomicité : une transaction doit fonctionner comme une seule unité de travail indivisible afin que la transaction entière soit appliquée ou annulée. Lorsque les transactions sont atomiques, il n'y a pas de transaction partiellement terminée: c'est tout ou rien.

- Cohérence : la base de données doit toujours passer d'un état cohérent à l'autre.

- Isolation : les résultats d'une transaction sont généralement invisibles pour les autres transactions jusqu'à ce que la transaction soit terminée.

- Durabilité : une fois validées, les modifications d'une transaction sont permanentes. Cela signifie que les modifications doivent être enregistrées de manière que les données ne soient pas perdues en cas de panne du système.

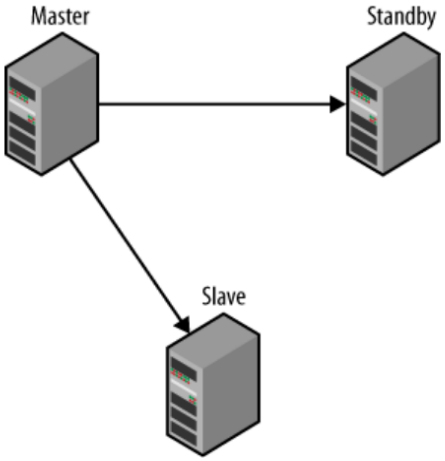
8. Gestion de la reprise sur panne

La journalisation des transactions permet de rendre les transactions plus efficaces. Au lieu de mettre à jour les tables sur le disque chaque fois qu'une modification se produit, le moteur de stockage peut modifier sa copie en mémoire des données. Le moteur de stockage peut alors écrire un enregistrement de la modification dans le journal des transactions, qui est sur disque et donc durable. Il s'agit également d'une opération relativement rapide, car l'ajout d'événements de journal implique des E / S séquentielles dans une petite zone du disque au lieu d'E / S aléatoires à de nombreux endroits. Ensuite, à un moment ultérieur, un processus peut mettre à jour la table sur le disque.

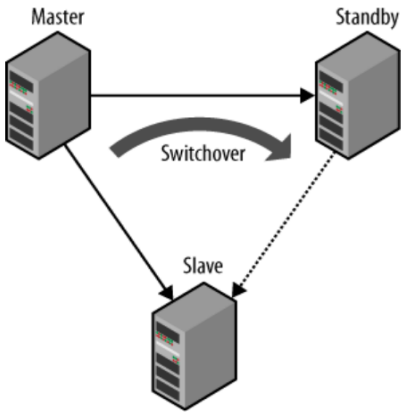
En cas de panne après l'écriture de la mise à jour dans le journal des transactions, mais avant que les modifications ne soient apportées aux données elles-mêmes, le moteur de stockage peut toujours récupérer les modifications au redémarrage. La méthode de récupération varie selon les moteurs de stockage.

La reprise sur panne garantit le retour au dernier état cohérent de la base précédant l'interruption, mais c'est au programmeur de définir ces points de cohérence (ou points de sauvegarde) dans le code des programmes.

Le plus simple des topologies pour la duplication de serveurs est la topologie de reprise à chaud. Elle se compose du maître et d'un serveur dédié appelé hot standby qui duplique le maître principal. Le serveur de secours automatique est connecté au maître en tant qu'esclave, et il lit et applique toutes les modifications.



L'idée est que lorsque le maître principal tombe en panne, la reprise à chaud fournit une réplique fidèle du maître, et tous les clients et esclaves peuvent donc basculer sur la reprise à chaud et continuer à fonctionner.



La commande mysqlbackup , qui fait partie du composant MySQL Enterprise Backup, permet de sauvegarder une instance MySQL en cours d'exécution, y compris des InnoDBtables, avec une interruption minimale des opérations tout en produisant un instantané cohérent de la base de données. Lorsque mysqlbackup copie des InnoDBtables, les lectures et écritures sur les InnoDBtables 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 récupération ponctuelle. MySQL Enterprise Backup fait partie de l'abonnement MySQL Enterprise.

Les sauvegardes à froid sont des sauvegardes effectuées sur des données alors que la base de données est hors ligne et inaccessible aux utilisateurs. Étant donné que les sauvegardes sont effectuées lorsque la base de données est hors ligne, les sauvegardes à froid sont également souvent appelées sauvegardes hors ligne. Les sauvegardes à froid consomment souvent moins de ressources et comme aucune nouvelle donnée ne peut être ajoutée (la base de données est hors ligne), le processus de sauvegarde peut sauvegarder toutes les données en une seule fois. Les sauvegardes à froid, bien sûr, ont une limitation dans le fait que la base de données n'est pas accessible lorsque l'opération de sauvegarde est en cours.

Pour la reprise sur les pannes à froid, il est possible d'arrêter le serveur MySQL et effectuer une sauvegarde physique composée de tous les fichiers utilisés par InnoDBpour gérer ses tables. Il faut utiliser la procédure suivante :

- Effectuer un arrêt lent du serveur MySQL et s'assurer qu'il s'arrête sans erreur.
- Copier tous InnoDBles fichiers de données (ibdatafichiers et .ibdfichiers) dans un endroit sûr.
- Copier tous InnoDBles fichiers journaux (ib_logfilefichiers) dans un endroit sûr.
- Copier le my.cnfou les fichiers de configuration dans un endroit sûr.

9. Techniques d'indexation

Lorsqu'un index est créé sur une table, MySQL stocke cet index sous forme d'une structure particulière, contenant les valeurs des colonnes impliquées dans l'index. Cette structure stocke les valeurs triées et permet d'accéder à chacune de manière efficace et rapide.

10. Optimisation de requêtes

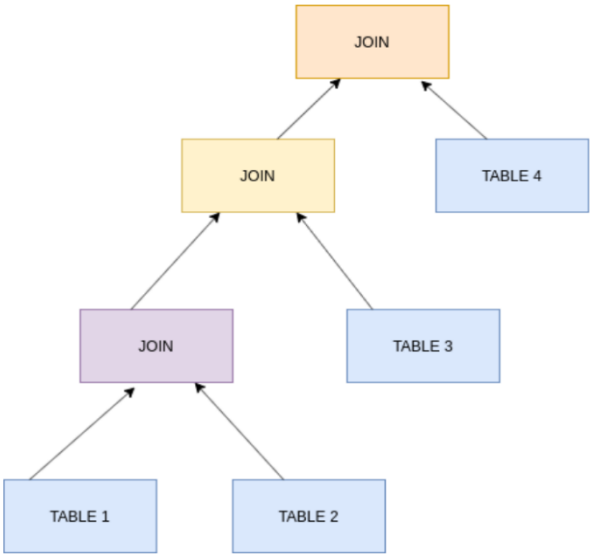
MySQL analyse les requêtes pour créer une structure interne (le parse tree).

Une fois que le client atteint l'exécution d'une requête, le serveur analyse d'abord la syntaxe de la requête. L'analyse syntaxique vérifie si les composants du langage sont correctement formés pour construire une instruction valide.

Ensuite, la signification de la requête syntaxiquement légale est évaluée, ce qui est connu sous le nom d' analyse sémantique . À ce stade, tous les objets de base de données de référence et les variables d'hôte seront vérifiés pour une utilisation précise. Une fois que la requête SQL écrite a réussi les deux vérifications, le serveur lui attribuera un ID SQL et une valeur de hachage (MD5). La valeur de hachage est basée sur les quelques centaines de caractères de l'instruction, de sorte que des collisions de hachage peuvent se produire, en particulier pour les instructions longues.

La prochaine étape consistera à analyser la requête dans un arbre d'analyse. La requête est exposée à une zone de mémoire partagée (pool partagé) qui contient des arbres d'analyse et des plans d'exécution pour les requêtes SQL exécutées. Chaque requête SQL unique posséderait une zone SQL partagée unique. L'analyse peut se produire de 2 manières principales par la suite.

Le plan d'exécution de la requête agit comme un arbre d'instructions que le moteur d'exécution de requête doit suivre afin de produire les résultats de la requête. L'une des optimisations les plus importantes du plan d'exécution est l'optimisation des jointures.



L'étape suivante est l'étape d'exécution des requêtes. MySQL suit simplement les instructions données dans le plan d'exécution de la requête.

La dernière étape de l'exécution de la requête consiste à envoyer le jeu de résultats au client, événement en présence d'un jeu de résultats nul.

L’optimiseur est l'ensemble des routines qui décident du chemin d'exécution que le SGBD doit emprunter pour les requêtes.

---------- *Fin du document* ----------

PARTIE 2:PROJET SUR LES THÉMATIQUES LIÉES À L'INDEXATION ET LA RECHERCHE

Voir le GitHub : https://github.com/MorganeU/ProjetSGBD.gi

PARTIE 3 (BONUS) : EXERCICES

Voir le GitHub : https://github.com/MorganeU/ProjetSGBD.gi

BIBLIOGRAPHIE

https://openclassrooms.com/fr/courses/1959476-administrez-vos-bases-de-donnees-avec-mysql/1962880-index

http://www.lire-fichier.com/logiciels/mysql#:~:text=Contient%20les%20informations%20de%20formatage,sont%20sauvegard%C3%A9s%20avec%20un%20fichier%20.

https://dev.mysql.com/doc/internals/en/optimizer-primary-optimizations.html

https://medium.com/analytics-vidhya/explore-your-recursive-sql-query-f15b120e518e

https://www.oreilly.com/library/view/high-performance-mysql/9781449332471/ch01.html

http://jpbeconne.free.fr/spip/spip.php?article12

https://stph.scenari-community.org/bdd/0/co/sql0c00.html

http://lacl.u-pec.fr/dima/bd/bd1.pdf

https://dev.mysql.com/doc/refman/8.0/en/innodb-buffer-pool.html#:~:text=The%20buffer%20pool%20is%20an,assigned%20to%20the%20buffer%20pool.

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

http://sys.bdpedia.fr/transactions.html

https://www.oreilly.com/library/view/mysql-high-availability/9780596807313/ch04.html

https://backup.ninja/news/hot-warm-and-cold-backups-mysql