MySQL, administration avancée

Dr-Ing Pierre-Emmanuel Gros
pierre-emmanuel.gros@neuresys.fr

Objectifs Pédagogiques

- Installer plusieurs instances de MySQL sur un même serveur
- Connaître les meilleures pratiques pour améliorer le stockage, comme la compression de tables, la défragmentation
- Améliorer la sécurité en utilisant SSL pour chiffrer les connexions d'utilisateurs
- Comprendre le principe de la réplication et la mettre en œuvre dans MySQL
- Mettre en place une architecture MySQL en cluster

Sommaire

- Rappels
- Fonctions avancées de l'administration
- Réplication
- MySQL cluster

Rappels

- Architecture MySQL et différents moteurs de stockage.
- Moteurs de stockage et plug-ins.
- Moteurs de stockage et index.
- Paramétrage du serveur MySQL.
- Paramétrage et gestion du moteur InnoDB.
- Collecte des statistiques.
- Journaux MySQL.

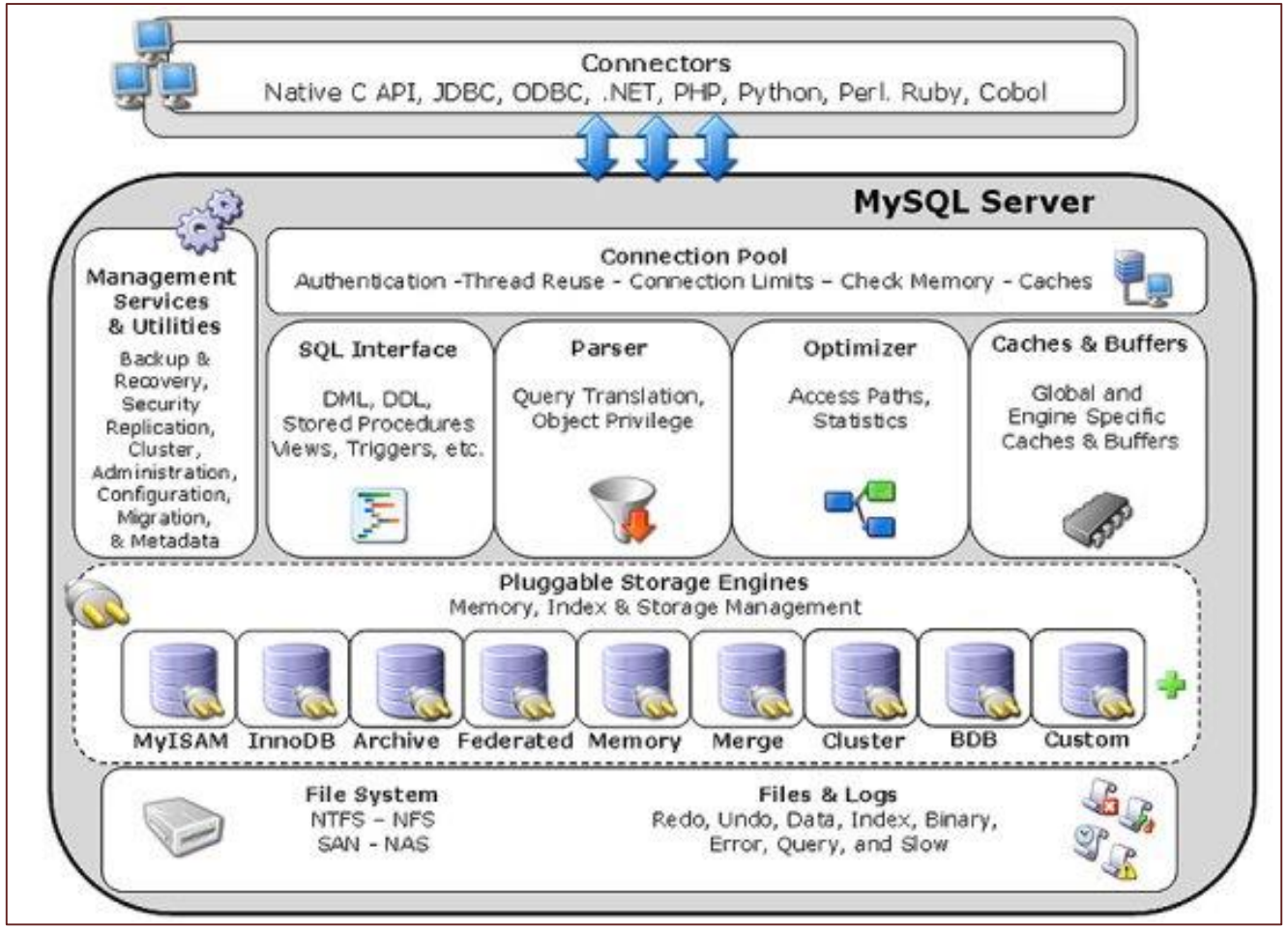
Architecture MySQL et différents moteurs de stockage

- Montrer rapidement l'architecture de MySQL au regard des moteurs de stockage.
- Voir les différences entre les différents moteurs de stockages InnoDB et MyISAM.
- Créer des tables avec différents moteurs.

Moteurs de stockage MyISAM, InnoDB

- Une particularité de MySQL est de disposer de moteurs de stockages modulaires.
- Chaque moteur possède des caractéristiques particulières (performance, transactions, recherche full text).
- Un serveur MySQL se charge de créer un certain nombre de thread:
  - Un thread global est responsable de la création et de gestion des connexions
  - Un thread surveilles les connexions et les termine.
  - Un thread pour la réplication, et un thread dédié à la sauvegarde des donnée sur le disque.

Plugable Storage Engine



Plugable Storage Engine

- Les moteurs de stockage sont des composants MySQL qui gèrent les opérations SQL pour différents types de tables. Les moteurs de stockage MySQL incluent à la fois ceux qui gèrent les tables sécurisées pour les transactions et ceux qui gèrent les tables non sécurisées pour les transactions. InnoDB est le moteur de stockage par défaut à partir de MySQL 5.5.5 (L'instruction CREATE TABLE dans MySQL 5.5 crée les tables InnoDB par défaut.)
- MySQL utilise une architecture de moteur de stockage enfichable qui permet aux moteurs de stockage d'être chargés et déchargés d'un serveur MySQL en cours d'exécution.
- Pour déterminer les moteurs de stockage pris en charge par votre serveur, utilisez l'instruction SHOW ENGINES. La valeur dans la colonne Support indique si un moteur peut être utilisé. Une valeur de YES, NO ou DEFAULT indique qu'un moteur est disponible, indisponible ou disponible et actuellement défini comme moteur de stockage par défaut.

SHOW ENGINES

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 2. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 3. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
```

Fichier FRM

- Quel que soit le moteur de stockage choisi, chaque table MySQL que vous créez est représentée sur le disque par un fichier .frm décrivant le format de la table. Le fichier porte le même nom que la table, avec une extension .frm. Le format .frm est identique sur toutes les plates-formes.
  - mysql> CREATE TABLE table1 (column1 CHAR(5)) ENGINE=MYISAM COMMENT '*';
  - shell> ls table1.*
- Le fichier frm représente la structure de la table. Le fichier MYD (MyIsam Data) et MYI (MyISAM Index) sont propre au storage engine MyISAM

MyISAM

- MyISAM est basé sur l'ancien moteur de stockage ISAM (et n'est plus disponible).
- MyISAM est un moteur rapide avec une faible empreinte en mémoire (seul les index sont chargés en RAM)
- Son usage est envisageable sur des données non critiques du fait des écarts de performances entre le moteur InnoDB et le moteur MyISAM.

MyISAM

- Chaque table MyISAM est stockée sur le disque dans deux fichiers. Les fichiers ont des noms qui commencent par le nom de la table et ont une extension pour indiquer le type de fichier. Le fichier de données a une extension .MYD (MYData). Le fichier d'index a une extension .MYI (MYIndex). La définition de la table est stockée dans le dictionnaire de données MySQL.
- Vous pouvez vérifier ou réparer les tables MyISAM avec le client mysqlcheck ou l'utilitaire myisamchk.
- Vous pouvez également compresser les tables MyISAM avec myisampack pour prendre beaucoup moins de place.
- CREATE TABLE t (i INT) ENGINE = MYISAM;

MyISAM fonctionnalités

- Prise en charge d'un véritable type VARCHAR: une colonne VARCHAR commence par une longueur stockée dans un ou deux octets.
- Non transactionnel, MyISAM ne garantit pas la fin d'une requête
- Seul les index sont en mémoire, et c'est l'OS qui garantit la mise en cache des données.
- Pas de support des clés étrangères

MyISAM caractéristiques techniques

- Toutes les valeurs de données sont d'abord stockées avec l'octet faible. Cela rend la machine de données et le système d'exploitation indépendants.
- Toutes les valeurs de clé numériques sont d'abord stockées avec l'octet de poids fort pour permettre une meilleure compression d'index
- Il y a une limite de (2 ^ 32) lignes dans une table MyISAM.
- Le nombre maximal d'index par table MyISAM est de 64.
- Le nombre maximal de colonnes par index est de 16. La longueur maximale de la clé est de 1000 octets.
- Lorsque les lignes sont insérées dans un ordre trié (comme lorsque vous utilisez une colonne AUTO_INCREMENT),
- Le traitement interne d'une colonne AUTO_INCREMENT par table est pris en charge.
- Les colonnes BLOB et TEXT peuvent être indexées.
- Les valeurs NULL sont autorisées dans les colonnes indexées. Cela prend 0 à 1 octet par clé.
- Chaque colonne de caractères peut avoir un jeu de caractères différent

My.ini exemple pour MyISAM

```
key_buffer_size                    = 8M
# Set to 25 - 33 % of RAM if you still use MyISAM
myisam_recover_options             = 'BACKUP,FORCE'
# Save a backup in case of recovering table and run it
```

Support technique

Fonctionnalités	Support
Index B-tree	Oui
Sauvegarde / récupération à un moment donné	Oui
Prise en charge de la base de données de cluster	Non
Index clusterisé	Non
Données compressées	Oui
Caches de données	Non
Données cryptées	Oui
Support de clé étrangère	Non
Index de hachage	Non
Limites de stockage	256 To
Mise à jour des statistiques pour le dictionnaire de données	Oui

InnoDB

- InnoDB est le moteur par défaut sur MySQL. C'est un moteur transactionnel complet qui supporte la recherche full text.
- Ce moteur est légèrement moins rapide que MyISAM.
- Il mobilise plus de ressources puisque les tables InnoDB doivent être chargées en mémoire vive pour fonctionner de manière optimale.

InnoDB

InnoDB est le moteur de stockage par défaut les raisons suivantes:

- Si le serveur tombe en panne, les tables InnoDB peuvent être rechargé sans problème.
- Le moteur de stockage InnoDB maintient son propre pool de mémoire tampon qui met en cache la table et indexe les données dans la mémoire principale lors de l'accès aux données.
- Si vous partionnez des données liées en différentes tables, vous pouvez configurer des clés étrangères qui appliquent l'intégrité référentielle.
- Optimisation des requêtes en utilisant les clef primaires (vue comme des index)
- Vous pouvez compresser des tables et des index associés. Vous pouvez créer et supprimer des index avec beaucoup moins d'impact sur les performances et la disponibilité.

InnoDB

Fonctionnalités	Support
Index B-tree	Oui
Sauvegarde / récupération à un moment donné	Oui
Prise en charge de la base de données de cluster	Non
Index clusterisé	Oui
Données compressées	Oui
Caches de données	Oui
Données cryptées	Oui
Support de clé étrangère	Oui
Index de hachage	Non (mise en place du Adaptative Hash Index)
Limites de stockage	64 To
Mise à jour des statistiques pour le dictionnaire de données	Oui

InnoDB

- Une table peut contenir un maximum de 1017 colonnes.
- Les colonnes générées virtuellement sont incluses dans cette limite.
- Une table peut contenir un maximum de 64 index secondaires.
- La longueur maximale du préfixe de la clé d'index est de 3 072 octets pour les tables InnoDB utilisant le format de ligne DYNAMIC ou COMPRESSED.
- La longueur maximale du préfixe de la clé d'index est de 767 octets pour les tables InnoDB utilisant le format de ligne REDUNDANT ou COMPACT.
  - Par exemple, vous pouvez atteindre cette limite avec un index de préfixe de colonne de plus de 191 caractères sur une colonne TEXT ou VARCHAR, en supposant un jeu de caractères utf8mb4 et un maximum de 4 octets pour chaque caractère. Toute tentative d'utilisation d'une longueur de préfixe de clé d'index supérieure à la limite renvoie une erreur.
- Les limites applicables aux préfixes de clé d'index s'appliquent également aux clés d'indexation de colonne complète. Si vous réduisez la taille de la page InnoDB à 8 ou 4 Ko en spécifiant l'option innodb_page_size lors de la création de l'instance MySQL, la longueur maximale de la clé d'index est réduite proportionnellement, en fonction de la limite de 3 072 octets pour une taille de page de 16 K
- Autrement dit, la longueur maximale de la clé d'index est de 1536 octets lorsque la taille de la page est de 8 Ko et de 768 octets lorsque la taille de la page est de 4 Ko. Un maximum de 16 colonnes est autorisé pour les index multicolonnes. Dépasser la limite renvoie une erreur.

Objets d'une base MySQL
Types de tables (MyISAM, MEMORY, MERGE...).

- Quels sont les types de tables?
- Découvertes des différents storage engine

Type de Table

MySQL fournit différents moteurs de stockage pour ses tables, comme suit:

- MyISAM
- InnoDB
- MERGE
- MEMORY (HEAP)
- ARCHIVE
- CSV
- FEDERATED

Chaque moteur de stockage a ses propres avantages et inconvénients. Il est essentiel de comprendre les fonctionnalités de chaque moteur de stockage et de choisir celle qui convient le mieux à vos tables afin d'optimiser les performances de la base de données. Dans les sections suivantes, nous aborderons chaque moteur de stockage et ses fonctionnalités afin que vous puissiez choisir celui à utiliser.

MyISAM

- MyISAM étend l'ancien moteur de stockage ISAM. Les tables MyISAM sont optimisées pour la compression et la vitesse. Les tables MyISAM sont également portables entre les plates-formes et les systèmes d'exploitation.
- La taille de la table MyISAM peut atteindre 256 To. De plus, les tables MyISAM peuvent être compressées en tables en lecture seule pour économiser des espaces.
- Au démarrage, MySQL vérifie la corruption des tables MyISAM et les répare même en cas d'erreur.
- Les tables MyISAM ne sont pas sécurisées pour les transactions. Avant MySQL version 5.5, MyISAM était le moteur de stockage par défaut lors de la création d’un table sans moteur de stockage explicite
- À partir de la version 5.5, MySQL utilise InnoDB comme moteur de stockage par défaut.

Création d'une table MyISAM

```
CREATE TABLE fyi_isam ( id INTEGER PRIMARY KEY, title
VARCHAR(80), count INTEGER ) ENGINE = MYISAM;
```

InnoDB

Les tables InnoDB prennent entièrement en charge les transactions et les transactions conformes à ACID. Ils sont également optimaux pour la performance.

La table InnoDB prend en charge les opérations de clés étrangères, de validation, d'annulation et de transfert.

La taille d'une table InnoDB peut atteindre 64 To. Comme MyISAM, les tables InnoDB sont portables entre différentes plates-formes et systèmes d'exploitation. MySQL vérifie et répare les tables InnoDB, si nécessaire, au démarrage.

Création d'une table InnoDB

```
CREATE TABLE fyi_isam ( id INTEGER PRIMARY KEY, title
VARCHAR(80), count INTEGER );
CREATE TABLE fyi_isam ( id INTEGER PRIMARY KEY, title
VARCHAR(80), count INTEGER ) ENGINE = INNODB;
```

MERGE

- Une table MERGE est une table virtuelle qui combine plusieurs tables MyISAM ayant une structure similaire à une table. Le moteur de stockage MERGE est également appelé moteur MRG_MyISAM.
- La table MERGE n'a pas ses propres index; il utilise plutôt les index des tables de composants.
- À l'aide de la table MERGE, vous pouvez accélérer les performances lorsque vous joignez plusieurs tables. MySQL vous permet uniquement d'effectuer des opérations SELECT, DELETE, UPDATE et INSERT sur les tables MERGE.
- Si vous utilisez l'instruction DROP TABLE sur une table MERGE, seule la spécification MERGE est supprimée. Les tables sous-jacentes ne seront pas affectées.

MERGE

```
mysql> CREATE TABLE t1 ( -> a INT NOT NULL AUTO_INCREMENT
PRIMARY KEY, -> message CHAR(20)) ENGINE=MyISAM;
mysql> CREATE TABLE t2 ( -> a INT NOT NULL AUTO_INCREMENT
PRIMARY KEY, -> message CHAR(20)) ENGINE=MyISAM;
mysql> INSERT INTO t1 (message) VALUES
('Testing'),('table'),('t1');
mysql> INSERT INTO t2 (message) VALUES
('Testing'),('table'),('t2');
mysql> CREATE TABLE total ( a INT NOT NULL AUTO_INCREMENT,
message CHAR(20), INDEX(a)) ENGINE=MERGE UNION=(t1,t2)
INSERT_METHOD=LAST;
```

MERGE

Les définitions et les index de la table sous-jacente doivent se conformer étroitement à la définition de la table MERGE.

La conformité est vérifiée lorsqu'une table faisant partie d'une table MERGE est ouverte, et non lorsque la table MERGE est créée.

Si une des tables échoue aux contrôles de conformité, l'opération ayant déclenché l'ouverture de la table échoue.

Cela signifie que les modifications apportées aux définitions des tables dans un MERGE peuvent provoquer un échec lors de l'accès à la table MERGE.

Les contrôles de conformité appliqués à chaque table sont les suivants:

- La table sous-jacente et la table MERGE doivent avoir le même nombre de colonnes.
- L'ordre des colonnes dans la table sous-jacente et dans la table MERGE doit correspondre.
- De plus, la spécification de chaque colonne correspondante de la table MERGE parent et des tables sous-jacentes est comparée et doit satisfaire à ces vérifications:
- Le type de colonne dans la table sous-jacente et la table MERGE doivent être égaux.
- La longueur de la colonne dans la table sous-jacente et la table MERGE doivent être égales.
- La colonne de la table sous-jacente et la table MERGE peuvent être NULL.
- La table sous-jacente doit avoir au moins autant d'index que la table MERGE.
- La table sous-jacente peut avoir plus d'indices que la table MERGE, mais ne peut en avoir moins.

MEMORY

Les tables de la mémoire sont stockées dans la mémoire et utilisent des index de hachage pour qu'ils soient plus rapides que les tables MyISAM.

La durée de vie des données des tables de la mémoire dépend de la disponibilité du serveur de base de données. Le moteur de stockage en mémoire est anciennement appelé HEAP.

MEMORY

Les performances de MEMORY sont limitées par les conflits résultant de **l'exécution d'un seul thread** et de la surcharge de verrouillage de table lors du traitement des mises à jour. Cela limite l'évolutivité lorsque la charge augmente, en particulier pour les mix d'instructions qui incluent des écritures.

Malgré le traitement en mémoire des tables MEMORY, elles ne sont pas nécessairement plus rapides que les tables InnoDB sur un serveur occupé, pour des requêtes à usage général ou sous une charge de travail en lecture / écriture.

En particulier, le verrouillage des tables impliqué dans les mises à jour peut ralentir l'utilisation simultanée des tables MEMORY à partir de plusieurs sessions. Selon les types de requêtes exécutées sur une table MEMORY, vous pouvez créer des index en tant que structure de données de hachage par défaut (pour rechercher des valeurs uniques basées sur une clé unique) ou en tant que structure de données B-tree à usage général (pour tous les types).

MEMORY

```
CREATE TABLE lookup (id INT, INDEX USING HASH (id)) ENGINE
= MEMORY;
CREATE TABLE lookup (id INT, INDEX USING BTREE (id)) ENGINE
= MEMORY;
```

MEMORY et Replication

Les tables MEMORY d'un serveur deviennent vides quand il est arrêté et redémarré. Si le serveur est un maître de réplication, ses esclaves ne sont pas conscients que ces tables sont devenues vides. Le contenu est donc obsolète si vous sélectionnez des données dans les tables des esclaves.

Archive

Le moteur de stockage d'archives vous permet de stocker un grand nombre d'enregistrements dans un format compressé à des fins d'archivage afin d'économiser de l'espace disque. Le moteur de stockage d'archives compresse un enregistrement lorsqu'il est inséré et le décompresse à l'aide de la bibliothèque zlib au fur et à mesure de sa lecture. Les tables d'archives n'autorisent que les instructions INSERT et SELECT. Les tables ARCHIVE ne supportent pas les index, une analyse complète de la table est donc nécessaire pour lire les lignes.

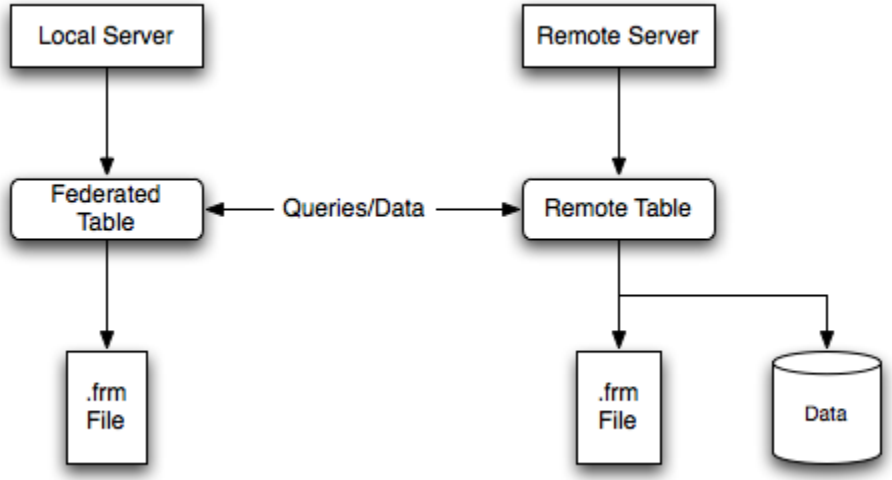
Une utilité des tables archives est de stocké des informations semblables en grand nombre (table de log par exemple)

CSV

Le moteur de stockage CSV stocke les données au format de fichier CSV (valeurs séparées par des virgules). Un tableau CSV constitue un moyen pratique de migrer des données vers des applications non SQL, telles que des tableurs. La table CSV ne prend pas en charge le type de données NULL. De plus, l'opération de lecture nécessite une analyse complète de la table.

FEDERATED

Le moteur de stockage FEDERATED vous permet de gérer les données d'un serveur MySQL distant sans utiliser la technologie de cluster ou de réplication. La table fédérée locale ne stocke aucune donnée. Lorsque vous interrogez des données à partir d'une table fédérée locale, les données sont extraites automatiquement des tables fédérées distantes.



FEDERATED

Pour créer une table FEDERATED, procédez comme suit:

- Créez la table sur le serveur distant. Vous pouvez également noter la définition de la table d'une table existante, en utilisant éventuellement l'instruction SHOW CREATE TABLE.
- Créez la table sur le serveur local avec une définition de table identique, mais en ajoutant les informations de connexion qui lient la table locale à la table distante.

FEDERATED TABLE via Connection

```
CREATE TABLE federated_table ( id INT(20) NOT NULL
AUTO_INCREMENT, name VARCHAR(32) NOT NULL DEFAULT '', other
INT(20) NOT NULL DEFAULT '0', PRIMARY KEY (id), INDEX name
(name), INDEX other_key (other) ) ENGINE=FEDERATED
DEFAULT CHARSET=utf8mb4
CONNECTION='mysql://fed_user@remote_host:9306/federated/tes
t_table';
```

FEDERATED via Server

```
CREATE SERVER fedlink FOREIGN DATA WRAPPER
mysql OPTIONS (USER 'fed_user', HOST
'remote_host', PORT 9306, DATABASE
'federated');


CREATE TABLE test_table ( id INT(20) NOT
NULL AUTO_INCREMENT, name VARCHAR(32) NOT
NULL DEFAULT '', other INT(20) NOT NULL
DEFAULT '0', PRIMARY KEY (id), INDEX name
(name), INDEX other_key (other) )
ENGINE=FEDERATED DEFAULT CHARSET=utf8mb4
CONNECTION='fedlink/test_table';
```

FEDERATED

Les éléments suivants indiquent les fonctionnalités que le moteur de stockage FEDERATED prend en charge et ne prend pas en charge:

- Le serveur distant doit être un serveur MySQL.
- La table distante pointée sur une table FEDERATED doit exister pour que vous puissiez accéder à la table par le biais de la table FEDERATED.
- Il est possible qu'une table FEDERATED en pointe une autre, mais veillez à ne pas créer de boucle.
- Une table FEDERATED ne supporte pas les index au sens habituel. comme l'accès aux données de la table est géré à distance, c'est la table distante qui utilise les index. **Cela signifie que, pour une requête ne pouvant utiliser aucun index et nécessitant une analyse complète de la table, le serveur extrait toutes les lignes de la table distante et les filtre localement.** Cela se produit indépendamment de tout WHERE ou LIMIT utilisé avec cette instruction SELECT; ces clauses sont appliquées localement aux lignes renvoyées. Les requêtes qui n'utilisent pas les index peuvent donc entraîner de mauvaises performances et une surcharge du réseau. **De plus, étant donné que les lignes renvoyées doivent être stockées en mémoire, une telle requête peut également entraîner le swap, voire l'interruption du serveur local.**

Résumé

Fonctionnalités	Innodb	MyISAM	Memory	CSV	Archive
B-tree					
Backup Recovery					
Clustering de données					
Clustering d'index					
Compression					
Cache de données					
Encryption					
Foreign Key					
Full Text					
Hash Index					
Type de Lock	Par ligne	Par table	Par table	Par table	Par ligne
MVCC (Transactionel)					
Replication					
Limit	64TB	256 TB	Mémoire	Disque	Disque
Transaction					

Magic Quadrant

Moteur de recherche - NDBCluster

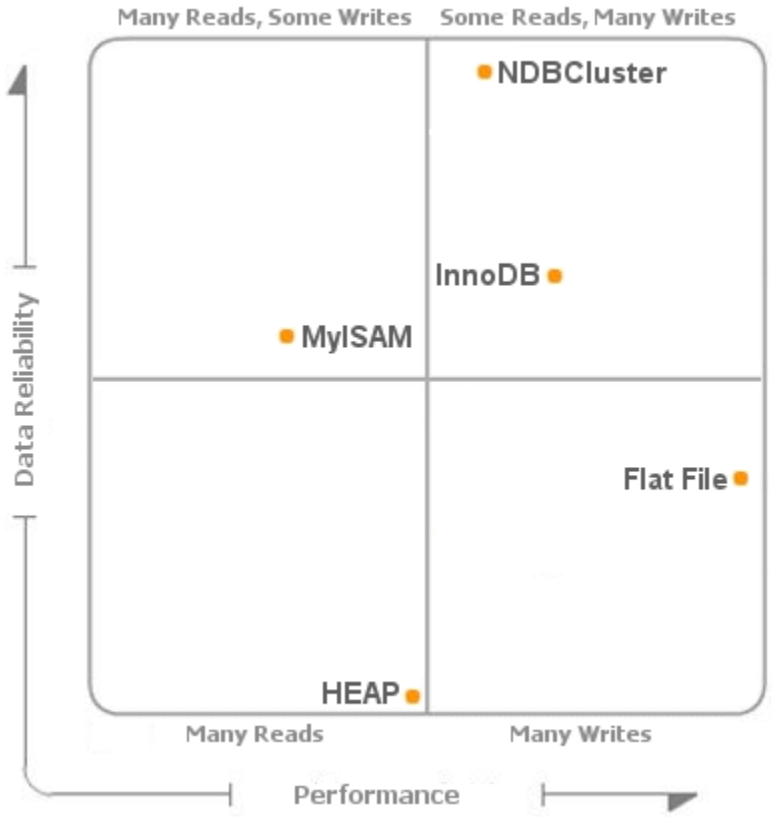
Journalisation des statistiques Web - Fichier plat pour la journalisation avec un démon de traitement hors ligne traitant et écrivant toutes les statistiques dans les tables InnoDB.

Transactions financières - InnoDB

Données de session - MyISAM ou NDBCluster

Calculs localisés - HEAP

Dictionnaire - MyISAM



Journaux

- Les journaux MySQL sont primordiaux. En effet, ils permettent de savoir les actions qui ont été effectuées avec succès ou non sur la base de données. Cela peut aussi permettre de détecter une erreur lorsque l'administrateur de la base de données ne la trouve pas.

Activation general_log

- Dans le fichier my.ini il faut activer le general_log et donner une destination

```
 GNU nano 2.2.6                    Fichier : /etc/mysql/my.cnf

#max_connections        = 100
#table_cache            = 64
#thread_concurrency     = 10
#
# * Query Cache Configuration
#
query_cache_limit       = 1M
query_cache_size        = 16M
#
# * Logging and Replication
#
# Both location gets rotated by the cronjob.
# Be aware that this log type is a performance killer.
# As of 5.1 you can enable the log at runtime!
general_log_file        = /var/log/mysql/mysql.log
general_log             = 1
#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
# Here you can see queries with especially long duration
slow_query_log_file = /var/log/mysql/mysql-slow.log
slow_query_log      = 1
long_query_time = 2
log_queries_not_using_indexes
```

Différents types « généraux » de logs

- general_log_file : directive précisant le fichier dans lequel seront stockés les journaux MySQL généraux
- general_log : directive permettant d'activer ou non la journalisation de MySQL
- slow_query_log_file : directive utilisée pour préciser le fichier dans lequel seront stockés les journaux MySQL concernant les requêtes lentes
- slow_query_log : directive permettant d'activer ou non la journalisation des requêtes lentes
- long_query_time : directive permettant une configuration de la durée en secondes au bout de laquelle MySQL considère la requête comme étant lente
- log_queries_not_using_indexes : directive permettant de préciser s'il faut ou non journaliser les requêtes qui n'utilisent pas d'index

Journal d'erreur

- Le paramètre log_error permet de signaler l'emplacement du journal des erreurs

```
  GNU nano 2.2.6                    Fichier : /etc/mysql/my.cnf

[client]
port            = 3306
socket          = /var/run/mysqld/mysqld.sock

# Here is entries for some specific programs
# The following values assume you have at least 32M ram

# This was formally known as [safe_mysqld]. Both versions are currently parsed.
[mysqld_safe]
log_error       = /var/log/mysql/mysql_err.log

[mysqld]
#
# * Basic Settings
#
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Error log - should be very few entries.
#
log_error = /var/log/mysql/error.log
#
```

Journal binaire

- Ce dernier est le plus important puisqu'il permet de journaliser toutes les requêtes DDL (Data Definition Language) et DML (Data Manipulation Language) effectuées sur le système de manière beaucoup plus claire que les journaux généraux.

```
 GNU nano 2.2.6                  Fichier : /etc/mysql/my.cnf

# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
expire_logs_days        = 10
max_binlog_size         = 500M
binlog_do_db            = include_database_name
binlog_ignore_db        = include_database_name
```

Journal binaire

Les paramètres les plus importants sont :

- log_bin : ce paramètre précise dans quel fichier seront stockés les journaux binaires
- expire_logs_days : paramètre permettant de définir la durée en jours avant expiration des journaux
- max_binlog_size : paramètre définissant la taille maximale du fichier de journaux binaires (au-delà de cette taille, MySQL incrémentera le nom du fichier pour en créer un nouveau)

L'outil mysqlbinlog permet de lire une log binaire et d'en avoir du sql.

Fonctions avancées de l’administration

- Structure interne des tables MyISAM. Compression de tables MyISAM.
- Défragmentation de l'espace des tables MyISAM. Réparation de tables MyISAM.
- Structure interne des tables transactionnelles InnoDB.
- Organisation de l'espace des tables InnoDB.
- Organisation des lignes dans les tables InnoDB.
- Fonctionnement transactionnel du moteur InnoDB.
- Connexions d'utilisateurs en SSL.

Fonctions avancées de l'administration
Compression de tables MyISAM.

- Présentation de myisampack afin de compresser les tables MyISAM

myisampack

- myisampack est un outil de compression de tables MyISAM. Les tables résultantes sont en lecture seule et généralement plus petites d'environ 40% à 70%. Il est exécuté comme suit:
- myisampack [options] file_name [file_name2...]
- Veuillez noter les points suivants:
  - Si le serveur mysqld a été appelé avec le verrouillage externe désactivé, ce n'est pas une bonne idée d'appeler myisampack si la table peut être mise à jour par le serveur.
  - Il est plus sûr de compresser les tables lorsque le serveur est arrêté.
  - **Après avoir compressé une table, celle-ci devient en lecture seule.**
  - myisampack ne supporte pas les tables partitionnées.

myisampack

```
shell> ls -l station.*
-rw-rw-r--   1 monty    my         994128 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          53248 Apr 17 19:00 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm
shell> myisampack station.MYI
Compressing station.MYI: (1192 records)
- Calculating statistics

normal:     20  empty-space:      16  empty-zero:        12  empty-fill:  11
pre-space:   0  end-space:        12  table-lookups:      5  zero:         7
Original trees:  57  After join: 17
- Compressing file
87.14%

shell> ls -l station.*
-rw-rw-r--   1 monty    my         127874 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          55296 Apr 17 19:04 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm
```

MyIsamPack

Les informations affichées par myisampack sont décrites ici :

- normal Le nombre de colonnes pour lesquelles aucune compression n'est utilisée.
- empty-space Le nombre de colonnes dont les valeurs ne contiennent que des octets : elles n'occuperont plus qu'un octet.
- empty-zero Le nombre de colonnes dont les valeurs ne contiennent que des zéros : elles n'occuperont plus qu'un octet.
- empty-fill Le nombre de colonnes de type entier qui n'occupent pas la totalité de l'espace de leur type. Elles seront réduites en taille (par exemple, une colonne de type INTEGER sera transformée en MEDIUMINT).
- pre-space Le nombre de colonnes de nombres à virgule flottante qui ont des valeurs stockées avec des espaces initiaux. Dans ce cas, chaque valeur va contenir le nombre d'espace initiaux. end-space Le nombre de colonnes qui ont de nombreux espaces terminaux. Dans ce cas, chaque valeur va contenir un compte du nombre d'espaces terminaux.
- Table-lookup La colonne n'a que quelques valeurs différentes, qui seront converties en une colonne de type ENUM avant une compression de type Huffman.
- Zero Le nombre de colonnes pour lesquelles toutes les valeurs sont zéro.
- Original trees Le nombre initial d'arbres Huffman.
- After join Le nombre d'arbres Huffman distincts obtenus après avoir joint les arbres pour économiser de l'espace d'entête.

Re-création des Index

Après la compression de myisampack vous devez exécuter la commande myisamchk pour recréer l'index. A ce moment, vous pouvez aussi trier les blocs d'index et créer des statistiques nécessaires pour l'optimiseur MySQL :

shell> myisamchk -rq --sort-index --analyze tbl_name.MYI

Fonctions avancées de l'administration
Défragmentation de l'espace des tables MyISAM. Réparation de tables MyISAM

- Pourquoi défragmenter les tables MyISAM.
- Présentation de l'instruction Optimize
- Présentation de myisamcheck

Pourquoi défragmenter?

- Sur MySQL**,** lorsque vous supprimez des enregistrements l'espace de l'enregistrement n'est pas réellement libéré
- Le problème avec ceci est que si une table effectuer de nombreuses opérations **DELETE,** l'espace physique de la table va devenir de plus en plus **fragmenté** et réduit la performance.
- Dans *MyISAM* et *InnoDB,* **OPTIMIZE TABLE** commande doivent exécuter sur l'optimisation de table, entre autres, fait une **défragmentation** automatique de la table.
- Il est fortement recommandé d'utiliser cette commande régulièrement en particulier sur les tables qui sont plus des déclarations de la suppression des enregistrements.

Défragmentation basé sur la performance

- L'un des symptômes de la fragmentation est qu'une table prend plus de place que nécessaire.
- Combien c'est exactement, est difficile à déterminer. Toutes les données et tous les index InnoDB sont stockés dans des arbres B et leur facteur de remplissage peut varier de 50% à 100%. Un autre symptôme de la fragmentation est qu'une analyse de table telle que celle-ci prend plus de temps qu'elle «ne devrait»

Défragmentation basé sur la taille

```
mysql> select table_name,
round(data_length/1024/1024) as data_length_mb,
round(data_free/1024/1024) as data_free_mb
 from information_schema.tables
 where round(data_free/1024/1024) > 500
 order by data_free_mb;

+------------+-----------------+---------------+
| table_name | data_length_mb | data_free_mb |
+------------+-----------------+---------------+
| BENEFITS   |            7743 |          4775 |
| DEPARTMENT |           14295 |         13315 |
| EMPLOYEE   |           21633 |         19834 |
+------------+-----------------+---------------+
```

Défragmentation basé sur la taille

Cela affichera la liste de toutes les tables ayant au moins 500 Mo d'espace inutilisé. Comme nous le voyons ci-dessus, dans cet exemple, il y a 3 tables qui ont plus de 500 Mo d'espace inutilisé.

- La colonne data_length_mb affiche la taille totale de la table en Mo. Par exemple, la taille de la table EMPLOYEE est d'environ 21 Go.
- La colonne data_free_mb affiche l'espace total inutilisé dans cette table particulière. Par exemple, la table EMPLOYEE contient environ 19 Go d'espace inutilisé.

Ces trois tables (EMPLOYEE, DEPARTEMENT ET AVANTAGES) sont très fragmentées et doivent être optimisées pour récupérer l'espace inutilisé.

Défragmentation basé sur la taille

- La taille du fichier sera la même que celle indiquée dans la colonne «data_length_mb» dans la sortie ci-dessus.

```
# ls -lh /var/lib/mysql/thegeekstuff/
..
-rw-rw----. 1 mysql mysql  7.6G Apr 23 10:55 BENEFITS.MYD
-rw-rw----. 1 mysql mysql   14G Apr 23 12:53 DEPARTMENT.MYD
-rw-rw----. 1 mysql mysql   22G Apr 23 12:03 EMPLOYEE.MYD
```

Défragmentation à l'aide de la commande OPTIMIZE TABLE

- La première méthode consiste à utiliser la commande Optimize table comme indiqué ci-dessous. L'exemple suivant optimisera la table EMPLOYEE.

```
mysql> OPTIMIZE TABLE EMPLOYEE, DEPARTMENT, BENEFITS
```

Optimize table

Quelques points à garder à l'esprit à propos d'optimiser le tableau:

- La table d'optimisation peut être effectuée pour le moteur InnoDB, le moteur MyISAM ou les tables ARCHIVE.
- Pour les tables MyISAM, il analysera la table, défragmentera le fichier de données MySQL correspondant et récupérera l'espace inutilisé.
- Pour les tables InnoDB, optimisation de table exécutera simplement une table de remplacement pour récupérer l'espace.
- Si vous avez des index, il utilisera également les pages d'index et mettra à jour les statistiques.
- Lors de l'optimisation, MySQL créera une table temporaire pour la table et, après l'optimisation, supprimera la table d'origine et renommera cette table temporaire.

Optimize Table

- Pour cet exemple, avant l'optimisation, vous verrez le fichier .MYD suivant pour la table. Lorsque la commande «OPTIMIZE TABLE» est en cours d'exécution, vous pouvez voir qu'elle a créé un fichier temporaire pour cette table avec l'extension .TMD.
- La taille de ce fichier temporaire continuera de croître jusqu'à ce que la table d'optimisation soit en cours d'exécution.
- Une fois la commande optimiser table terminée, vous ne verrez plus la table temporaire. Au lieu de cela, vous verrez le fichier EMPLOYEE.MYD d'origine optimisé et de taille réduite.

```
-rw-rw----. 1 mysql mysql   22G Apr 23 12:03 EMPLOYEE.MYD
Puis
-rw-rw----. 1 mysql mysql   22G Apr 23 12:03 EMPLOYEE.MYD
-rw-rw----. 1 mysql mysql  500M Apr 23 14:10 EMPLOYEE.TMD
Puis
-rw-rw----. 1 mysql mysql    2G Apr 23 14:20 EMPLOYEE.MYD
```

Défragmentation à l'aide de la commande mysqlcheck

- La deuxième méthode pour optimiser une table utilise la commande mysqlcheck comme indiqué ci-dessous. L'exemple suivant optimisera la table DEPARTMENT. Vous exécuterez cette commande à partir de l'invite Linux (et non sur l'invite MySQL).

```
# mysqlcheck -o thegeekstuff DEPARTMENT -u root -
pMySQLSecretPwd99
thegeekstuff.DEPARTMENT  OK
```

mysqlcheck

Remarque: en interne, la commande mysqlcheck utilise la commande “OPTIMIZE TABLE”.

Dans l'exemple ci-dessus: mysqlcheck est la commande qui est exécutée à partir de l'invite Linux.

- L'option -o indique que mysqlcheck doit effectuer l'opération «optimisation de la table».
- thegeekstuff est la base de données
- DEPARTMENT est la table à l'intérieur de la base de données geekstuff qui doit être optimisée
- -u root indique que la commande mysqlcheck doit utiliser «root» comme utilisateur mysql pour se connecter.
- -p indique le mot de passe du compte root de mysql. Veuillez noter qu'il n'y a pas d'espace entre l'option -p et le mot de passe.

Après Défragmentation

- Après l'optimisation, à l'aide de la requête suivante, vérifiez la taille totale et la taille de l'espace inutilisé pour les trois tables optimisées dans cet exemple.

```
mysql> select table_name,
round(data_length/1024/1024) as data_length_mb,
round(data_free/1024/1024) as data_free_mb
 from information_schema.tables
 where table_name in
 ( 'EMPLOYEE', 'DEPARTMENT', 'BENEFITS' );

+------------+----------------+--------------+
| table_name | data_length_mb | data_free_mb |
+------------+----------------+--------------+
| BENEFITS   |           2968 |            0 |
| DEPARTMENT |            980 |            0 |
| EMPLOYEE   |           1799 |            0 |
+------------+----------------+--------------+
```

Fonctions avancées de l'administration
Structure interne des tables transactionnelles InnoDB

- Présentation interne du moteur InnoDB.
- Apprentissage d'un l'intérieur d'un moteur de bases de données

InnoDB

Le moteur InnoDB a été conçue comme:

- OLTP : moteur effectuant des modifications d'information en temps réel.
- Performant, Stable et capable de montée en charge
- Capable de résister aux crashs
- Portable

Fonctionnalités InnoDB

Le moteur est conçue ainsi:

- Complètement transactionnel (commit, rollback)
- Pouvant mettre un lock par ligne
- Efficace en Entrée/Sortie
- MVCC (Multiversion concurrency control) : La base de données ne mettra pas en œuvre des mises à jour par écrasement des anciennes données par les nouvelles, mais plutôt en indiquant que les anciennes données sont obsolètes et en ajoutant une nouvelle "version". Ainsi, plusieurs versions sont stockées, dont l'une seule d'entre elle est la plus récente.

Particularité:

- Adaptive Hash Index : Création d'un cache automatique base sur les prefix des clefs
- Change Buffer: Bufferisation des pages a modifier
- DoubleWrite Buffer: Buffer d'écriture de page avant écriture sur disque

Architecture Technique InnoDB

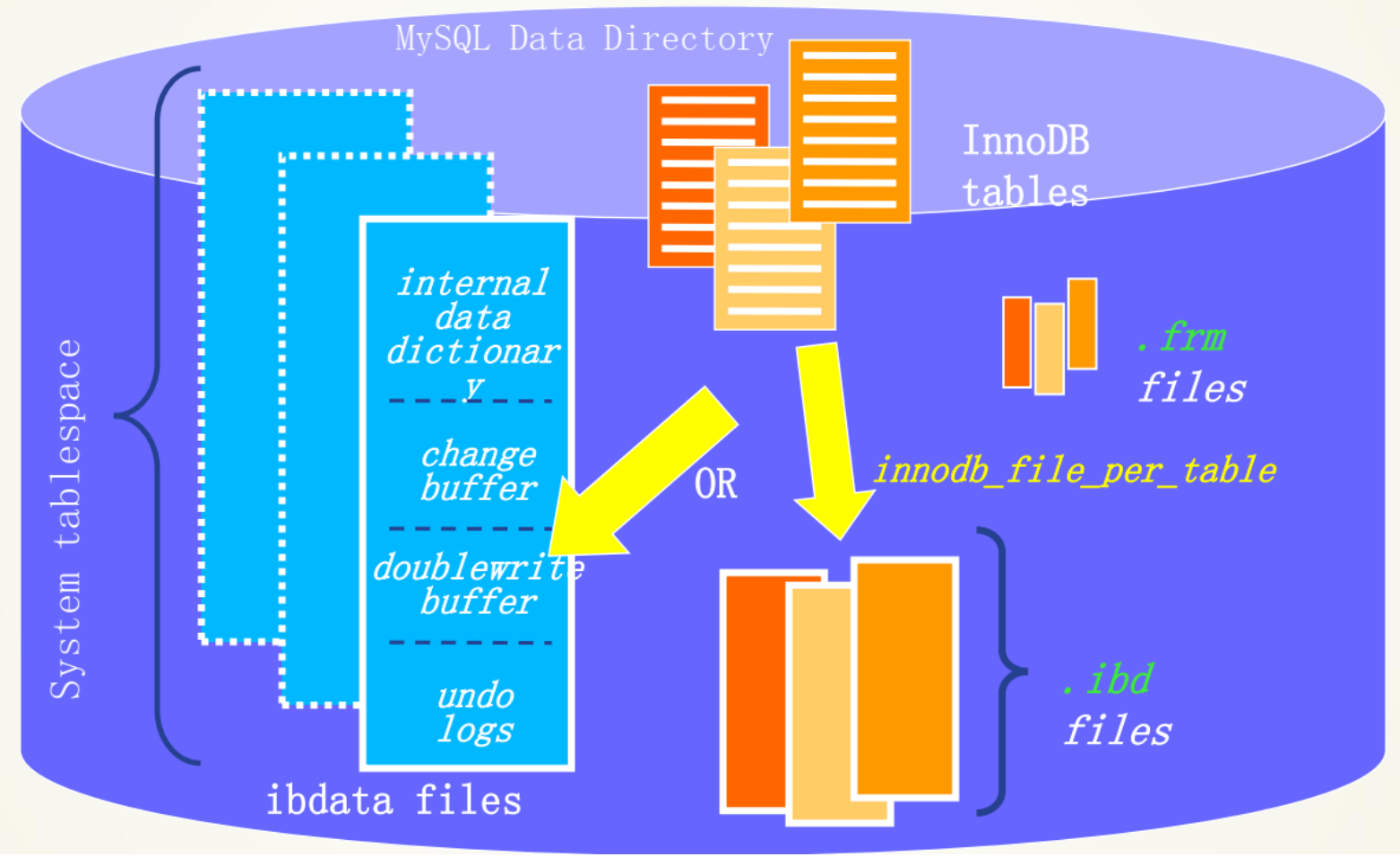
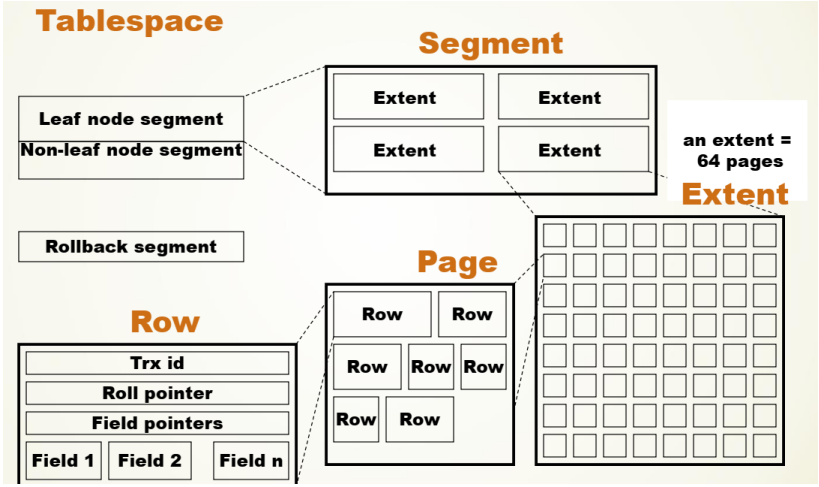
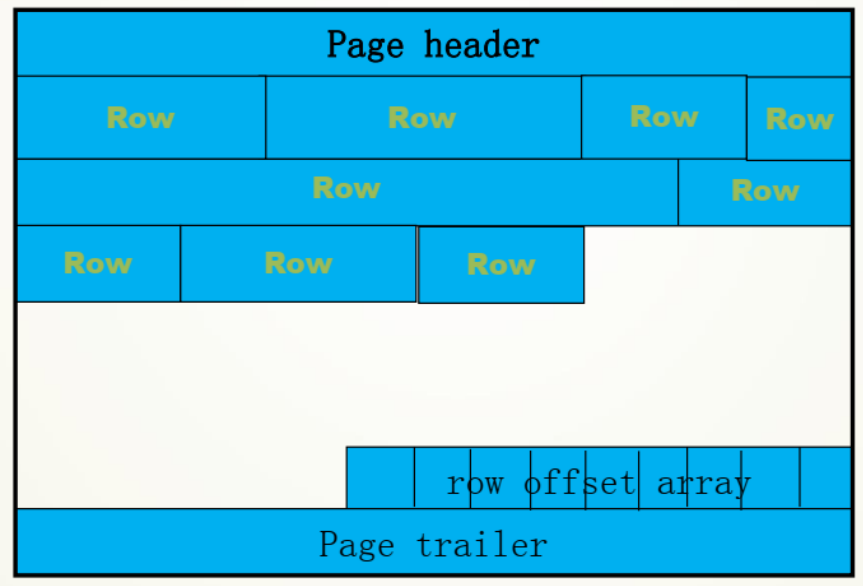


Table Space

- Le Table Space est l'organisation physique des données
- C'est la gestion des blob/clob dans les lignes qui implique le choix du format:
  - Dynamic, le field ne contient qu'un pointeur
  - Compress: les données sont compressé
  - Compact format par défaut





Problématique MVCC

- Faire du MVCC c'est être capable de faire des commit, rollback
- Autoriser en cas de crash a récupérer les données
- Ne pas écrire les données de façon ordonnées immédiatement.

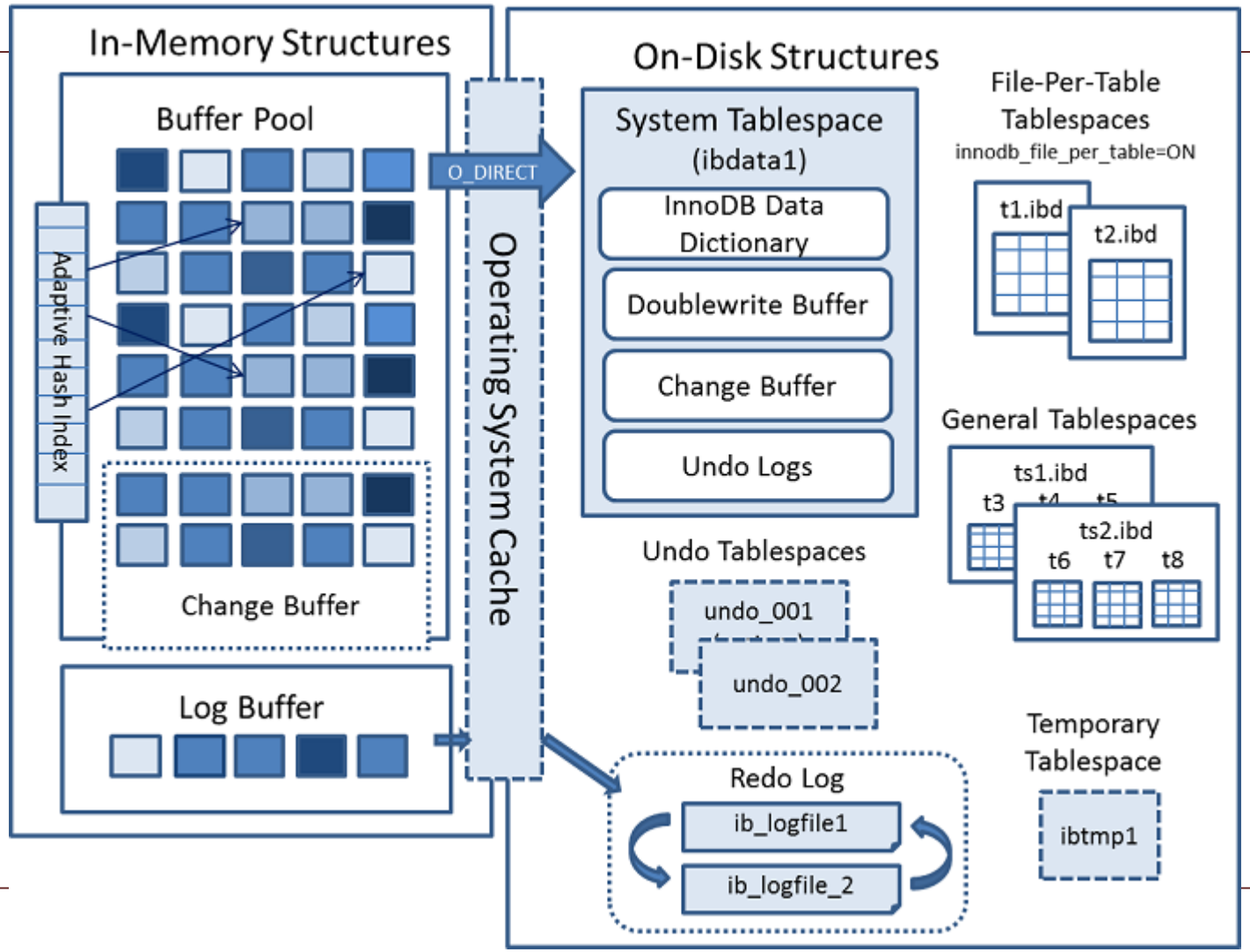
- Il y a donc besoin en sus de données des données de logs (modification en cours et modification a annuler).
  - Ces logs sont a conserver ne mémoire
  - Et a conserver sur disque (en cas de crash)

InnoDB Elements

- Chaque transaction en lecture / écriture doit avoir un ID de transaction unique.

- Vue de lecture: dans InnoDB, un instantané est utilisé pour une lecture cohérente.
- Redo Log : il est utilisé pour enregistrer les modifications apportées aux fichiers physiques. Toutes les modifications apportées aux fichiers physiques InnoDB doivent être protégées à l'aide de fichiers de journalisation pour faciliter la récupération après incident.
- Undo Log : il est utilisé pour stocker les journaux d'origine avant qu'ils ne soient modifiés.
- Binary Log: Log générique au format binaire de ce qui est fait afin de pouvoir faire une replication

Architecture Technique InnoDB

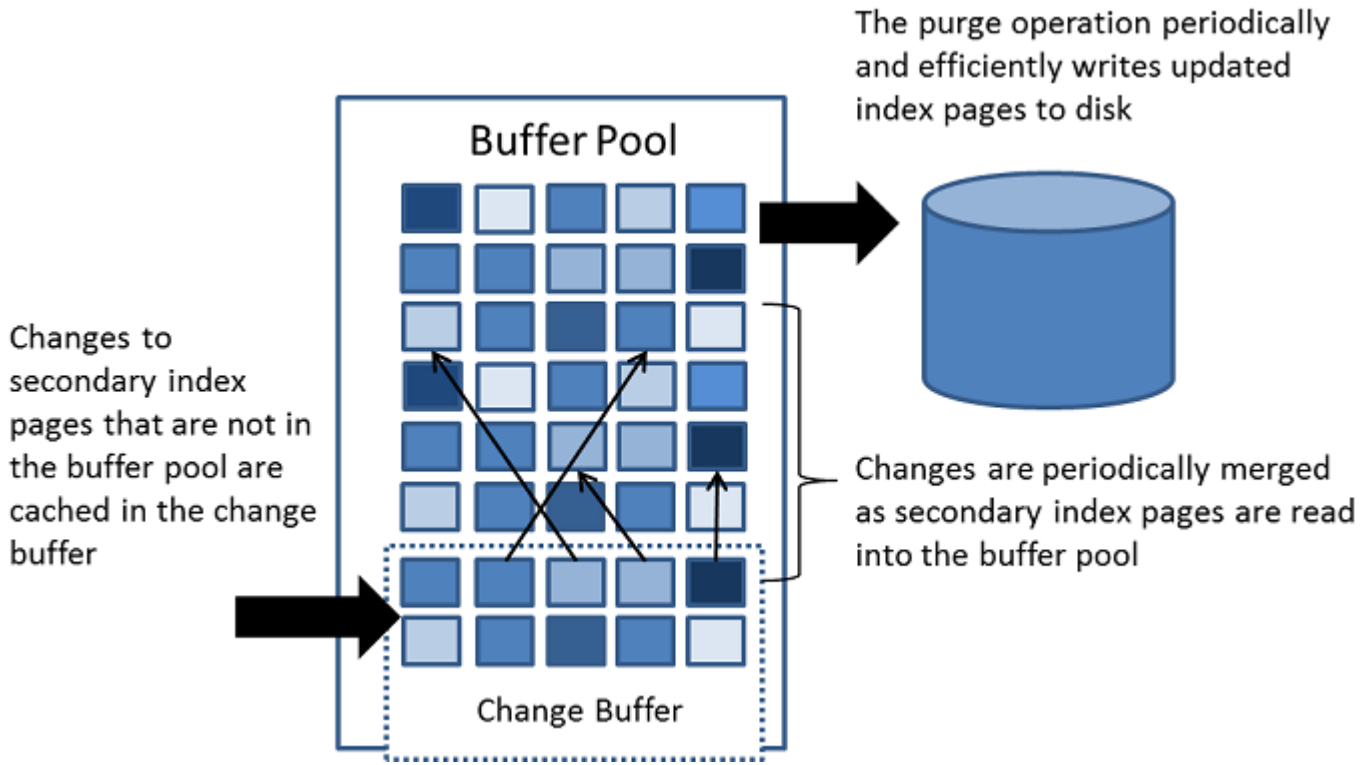


Buffer Pool

- Le Buffer Pool est une zone de la mémoire où sont mises en cache la table et les données d'index lors de l'accès. Le pool de mémoire tampon permet de traiter directement les données fréquemment utilisées à partir de la mémoire, ce qui accélère le traitement. Sur les serveurs dédiés, jusqu'à 80% de la mémoire physique est souvent affectée au pool de mémoire tampon.
- Pour que les opérations de lecture soient efficaces, Buffer Pool est divisé en pages pouvant potentiellement contenir plusieurs lignes.

Change Buffer

- Le Change Buffer permet met en cache les index secondaire lorsque ceux-ci ne sont pas dans le Buffer Pool



Adaptive Hash Index

- Si une table tient presque entièrement dans la mémoire principale, le moyen le plus rapide d'exécuter des requêtes consiste à utiliser des index de hachage plutôt que des recherches dans l'arbre B.
- InnoDB surveille les recherches sur chaque index défini pour une table. S'il constate que certaines valeurs d'index sont fréquemment utilisées, il crée automatiquement une table de hachage en mémoire pour cet index.
- Basé sur le modèle de recherche observé par InnoDB, il crée un index de hachage en utilisant un préfixe de la clé d'index.

Log Buffer

- Le log buffer est la zone de mémoire contenant les données à écrire dans les fichiers journaux du disque.

On Disk System Table Space

- Le system table space contient:
  - Le dictionnaire des données InnodB
  - Le double write buffer
  - Le change buffer
  - Le undo/redo log
- En particulier dans le répertoire datadir, vous pouvez voir le fichier idbdata1 par défaut.

On Disk InnoDB Data Dictionary

- Le dictionnaire de données InnoDB est composé de tables système internes contenant des métadonnées permettant de suivre des objets tels que des tables, des index et des colonnes de table. Les métadonnées sont physiquement situées dans l'espace de table système InnoDB.
- Pour des raisons historiques, les métadonnées du dictionnaire de données se superposent aux informations stockées dans les fichiers de métadonnées de tables InnoDB (fichiers .frm).

On Disk Double Write Buffer

- InnoDB utilise une zone réservée dans son espace de table principal, appelée tampon de Double Write Buffer, pour empêcher la corruption des données pouvant survenir avec des écritures de page partielles.
- Lorsque les données du pool de mémoire tampon sont vidées sur le disque, InnoDB purge les pages entières à la fois (par défaut, les pages de 16 Ko), et pas uniquement les enregistrements modifiés dans une page.
- Avec le Double Write Buffer, InnoDB / XtraDB écrit d'abord la page dans le tampon d'écriture double, puis dans les fichiers de données. Si une écriture de page partielle se produit dans les fichiers de données, InnoDB / XtraDB vérifiera la récupération si la somme de contrôle de la page du fichier de données est différente du Double Write Buffer et saura ainsi si la page est corrompue ou endommagée. ne pas.
- S'il est corrompu, le processus de récupération utilisera la page stockée dans le Double Write Buffer pour restaurer les données correctes.

On Disk Change Buffer

- Les instructions INSERT, UPDATE et DELETE peuvent être des opérations particulièrement lourdes car tous les index doivent être mis à jour après chaque modification. Pour cette raison, ces modifications sont souvent mises en mémoire tampon.
- Les pages sont modifiées dans le pool de mémoire tampon et non immédiatement sur le disque. Lorsque des lignes sont supprimées, un indicateur est défini. Par conséquent, les lignes ne sont pas immédiatement supprimées sur le disque

On Disk Redo Log

- Le Redo Log est une structure de données sur disque utilisée lors de la récupération sur incident pour corriger les données écrites par des transactions incomplètes.
- Lors des opérations normales, le Redo Log code les demandes de modification des données de table résultant d'instructions SQL ou d'appels d'API de bas niveau.
- Les modifications qui n'ont pas fini de mettre à jour les fichiers de données avant un arrêt inattendu sont automatiquement ré-exécutées lors de l'initialisation et avant que les connexions ne soient acceptées.
- Par défaut, le redo log est représenté physiquement sur le disque par deux fichiers nommés ib_logfile0 et ib_logfile1. MySQL écrit dans les fichiers de journalisation de manière circulaire.

On Disk Undo Log

- Avant qu'une ligne ne soit modifiée, elle est copiée dans le Undo Log. Chaque ligne contient un pointeur sur la version la plus récente de la même ligne dans le Undo Log. Chaque ligne du journal de rétablissement contient un pointeur sur la version précédente, le cas échéant. Ainsi, pour chaque ligne modifiée a ont a une chaîne d'historique.
- Le Undo Log n'est pas un fichier journal pouvant être visualisé sur le disque dans le sens habituel, tel que le journal des erreurs ou le journal de requête lent, plutôt qu'une zone de stockage.

On Disk Temporary Tablespace

- C'est l'espace des tables temporaires non compressées créées par l'utilisateur et les tables temporaires internes sur disque sont créées dans un espace de table temporaire partagé.

On Disk File Per Table TableSpaces

- Historiquement, les tables InnoDB étaient stockées dans l'espace de table système. Cette approche monolithique était destinée aux machines dédiées au traitement de bases de données, avec une croissance des données soigneusement planifiée, **où aucun stockage sur disque alloué à MySQL ne serait jamais nécessaire à d'autres fins**.
- La fonctionnalité de Per Table TableSpaces fournit une alternative plus flexible, dans laquelle chaque table InnoDB est stockée dans son propre fichier de données de tablespace (fichier .ibd).

On Disk General Tablespace

- Un espace de table général est un espace de table InnoDB partagé créé à l'aide de la syntaxe CREATE TABLESPACE.
- Il s'agit d'un mode d'utilisation entre le system table space et le per file system table space.
- La fonctionnalité de tablespace général offre les fonctionnalités suivantes:
  - Semblables à l'espace de table système, les espaces de table généraux sont des espaces de table partagés pouvant stocker des données pour plusieurs tables.
  - Les general table space présentent un avantage potentiel en termes de mémoire par rapport aux per file system table space

On Disk General Tablespace

- Les General Tablespace sont créés à l'aide de la syntaxe CREATE TABLESPACE.
- CREATE TABLESPACE nom_espace_table ADD DATAFILE 'nom_fichier' [FILE_BLOCK_SIZE = valeur] [ENGINE [=] nom_moteur]
- Un General Tablespace peut être créé dans le répertoire de données ou en dehors de celui-ci.
- Lors de la création d'un General Tablespace en dehors du répertoire de données, ce dernier doit exister avant la création de l'espace de table. Un fichier .isl est créé dans le répertoire de données MySQL lorsqu'un General Tablespace est créé en dehors du répertoire de données MySQL.

Creation d'un General Table Space

```
mysql> CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd'
Engine=InnoDB;
mysql> CREATE TABLESPACE `ts1` ADD DATAFILE
'/my/tablespace/directory/ts1.ibd' Engine=InnoDB;
```

Creation/Suppression

```
mysql> CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd'
Engine=InnoDB;
mysql> CREATE TABLE t1 (c1 INT PRIMARY KEY) TABLESPACE ts10
Engine=InnoDB;
mysql> DROP TABLE t1;
mysql> DROP TABLESPACE ts1;
```

Déplacement

Pour déplacer une table d'un Per File TableSpace ou d'un espace table système vers un General Tablespace.

ALTER TABLE nom_table TABLESPACE [=] nom_espace-table;

Pour déplacer vers le tablespace système

ALTER TABLE nom_table TABLESPACE [=] innodb_system;

Pour déplacer vers le per file tablespace

ALTER TABLE nom_de_table TABLESPACE [=] innodb_file_per_table;

Limitations

- La création Temporary Tablespace n'est pas prise en charge.
- Les General Tablespace ne prennent pas en charge les tables temporaires.
- Les tables stockées dans un General Tablespace ne peuvent être ouvertes que dans des versions de MySQL prenant en charge les espaces de table généraux.
- Comme pour le tablespace système, le fait de tronquer ou de supprimer des tables stockées dans un General Tablespace crée un espace libre interne dans le fichier de données .ibd du tablespace général, qui ne peut être utilisé que pour les nouvelles données InnoDB.
- L'espace n'est pas libéré dans le système d'exploitation, comme c'est le cas pour les espaces de table fichier par table.

Stockage sur le disque

innodb_file_format permet de signifier le format physique des fichiers InnoDB. Actuellement les formats Antelop et Barracuda sont supportés.

Ces formats de fichier permet de signifier comment sont stocké les données dans les fichiers: COMPACT, REDUNDANT, DYNAMIC ou COMPRESSED

ROW_FORMAT

- Utiliser DYNAMIC ou COMPRESSED signifie qu'InnoDB stocke les champs varchar / text / blob qui ne rentrent pas dans la page complètement hors page. Mais à part ces colonnes, qui ne comptent alors que 20 octets par colonne, la limite de taille de ligne InnoDB n'a pas changé. il est toujours limité à environ 8 000 octets par ligne.

ROW_FORMAT

- Le moteur InnoDB supporte 4 formats de stockage différents : les formats COMPACT, COMPRESSED et DYNAMIC. Ils peuvent être précisés avec l'option ROW FORMAT avec la clause CREATE TABLE. Par défaut, le format est COMPACT.
- Compact: Dans ce format les colonnes BLOB et TEXT sont partiellement stockées dans les pages de données. Au moins 767 octets sont stockés dans la ligne; tous les débords sont stockés dans des pages dédiées. Comme les tailles de lignes de Compact et Redundant sont d'environ 8000 octets, cela limite le nombre de colonnes BLOB ou TEXT qui peuvent être utilisées dans une table. Chaque page de BLOB page contient 16Ko, sans compter les données.
- Les autres colonnes peuvent être stockées dans des pages différentes si elles dépassent la taille de ligne maximum par page.

ROW_FORMAT

- Dynamic: Avec le format Dynamic, les colonnes BLOB et TEXT sont stockées différemment de Compact. Si les données ne peuvent être contenues dans une ligne de page, alors seulement un pointeur sera stocké et contiendra l'adresse de la page dédiée. Chaque page externe contient une partie des données et l'adresse de la page suivante, si nécessaire. Les pointeurs ont une taille de 20Ko. Cela permet de stocker un grand nombre de colonnes BLOB ou de TEXT dans une table.

ROW_FORMAT

- Compressed :Le format Compressed est une extension du format Dynamic permettant la réduction la taille des données

Configuration InnoDB

- Quelque élément de configuration InnoDB
- Focus sur les mémoires/io/consistence

Mémoire
innodb_buffer_pool_size

- Principal cache de mémoire global établi lors de l'initialisation de MySQL utilisé pour prendre en charge les opérations InnoDB. Il stocke principalement des données et des pages d'index, mais met également en cache d'autres informations telles que la mise en cache du journal, le tampon de changement, etc.
- **Defaut**: 134217728 (bytes) (128M)
- Dans la sortie SHOW ENGINE INNODB STATUS, dans la section BUFFER POOL AND MEMORY, vous verrez une entrée pour les mémoires tampons libres. Si la valeur est 0, vous savez qu'il n'y a plus d'espace dans le pool de mémoire tampon.

Mémoire
Innodb_buffer_pool_instances

- Le nombre d'instances de pools de mémoire tampon détermine le nombre de pools de mémoire tampon InnoDB individuels créés au démarrage de MySQL. Cela vise à réduire les conflits de caches individuels.
- Chaque instance de pool de mémoire tampon innodb doit être considérée comme une ‘région’ du pool de mémoire tampon InnoDB car elle divise la taille totale du pool de mémoire tampon InnoDB parmi le nombre d’instances spécifiées. IE: Si vous avez une taille de pool de mémoire tampon InnoDB de 8 Go et 8 instances de pool de mémoire tampon, chaque instance aura une taille de 1 Go.

Mémoire
Innodb_buffer_pool_instances

- Valeur par défaut: 8 (instances). Toutefois, si vous avez un pool de mémoire tampon inférieur à 1G, la valeur par défaut devient 1. Dans les versions précédentes de 5.6, la valeur par défaut était 1.
- Remarques sur le réglage: Si vous avez un grand pool de mémoire tampon InnoDB, vous devez l'accorder afin que vous disposiez de plusieurs instances. Toutefois, il convient de noter qu'un hachage est nécessaire pour prendre en charge le mappage des pages vers leur instance de pool de mémoire tampon. Ce hachage sera nécessaire quel que soit le nombre d'instances que vous possédez si le nombre d'instances est supérieur à 1. Je vous recommande de commencer par la valeur par défaut. Toutefois, si vous trouvez toujours un conflit en utilisant la métrique associée, essayez d'en ajouter davantage. C'est généralement une bonne idée de s'assurer qu'aucune instance n'est inférieure à 1 G en taille.

Mémoire
Innodb_old_blocks_pct

- Les pages se déplacent dans le pool de mémoire tampon InnoDB à l'aide de l'algorithme le moins récemment utilisé (LRU). Cependant, la liste des LRU du pool de mémoire tampon InnoDB comporte deux sections: les anciennes et les nouvelles sous-listes de pages.
- La nouvelle sous-liste de pages est placée au-dessus de l'ancienne sous-liste dans la liste LRU. Lorsqu'une nouvelle page est lue sur le disque et extraite dans le pool de mémoire tampon InnoDB, elle n'est pas ajoutée en haut de la liste des LRU. Elle est plutôt ajoutée à l'ancienne sous-liste des pages.
- À partir de ce moment, l'une des deux actions sera effectuée. Il sera soit relu tant qu'il existe encore dans le pool de tampons et sera déplacé en haut de la nouvelle sous-liste de pages (tout en haut de la liste des LRU), soit il ne sera pas relu et finira par atteindre le bas de la liste. les anciennes pages sous-liste et être expulsé du pool de mémoire tampon.
- Cette variable indique quelle quantité du pool de mémoire tampon InnoDB doit être utilisée par l'ancienne sous-liste de pages.

Mémoire
Innodb_old_blocks_pct

- Par défaut: 37 (pourcentage)
- Notes de réglage: La raison pour laquelle InnoDB n'ajoute pas les pages nouvellement lues en haut de la liste des LRU est de s'assurer qu'il y a de la place pour les pages chaudes (celles qui sont lues fréquemment) et que les pages moins fréquemment utilisées occupent moins de place. espace et également être expulsé du pool de mémoire tampon plus rapidement. Toutefois, vous risquez de voir les performances dégradées si de nombreuses pages sont expulsées avant qu'elles ne deviennent nouvelles, en particulier avec des pools de mémoire tampon InnoDB plus petits.

Mémoire
Innodb_old_blocks_time

- C'est la durée pendant laquelle une nouvelle page à partir d'un disque doit être stockée dans l'ancienne sous-liste de pages, quel que soit le nombre de fois où elle a été consultée. Une fois cette période écoulée, si la page se trouve toujours dans le pool de mémoire tampon et si vous y accédez à nouveau, elle se déplacera vers la nouvelle sous-liste de pages.
- Valeur par défaut: 1000 (millisecondes)
- Remarques sur le réglage: Cette variable est destinée à protéger le pool de mémoire tampon InnoDB contre le saturation complète des analyses de table complètes. Lorsqu'une analyse complète de la table est effectuée, le contenu d'une table entière sera déplacé dans le pool de mémoire tampon où il peut être lu un petit nombre de fois en succession rapide, puis ne jamais être lu à nouveau. Le fait de définir cette valeur sur 0 empêche les analyses de table de supprimer les pages de la nouvelle sous-liste de pages qui ne doivent pas être supprimées.

Mémoire
adaptive_hash_index

- InnoDB a un processus qui établit un profil des requêtes par rapport à l'utilisation de l'index sur les tables InnoDB. Sur la base de ses observations, il construira un index de hachage qui pointe vers les pages d'index fréquemment consultées en fonction du préfixe de la page d'index afin d'accélérer les recherches. Considérant qu'il s'agit d'un index de hachage, il ne peut pas être utilisé pour accélérer les recherches ou les commandes de plages, mais il peut considérablement augmenter les temps de recherche en recherchant ce point dans des enregistrements spécifiques tels que ceux qui utilisent =, <>, <=> et opérateurs IN.

Mémoire
adaptive_hash_index

- Par défaut: 1 (activé)
- Notes de réglage: vous avez très peu de contrôle sur le fonctionnement de l'index de hachage adaptatif ou sur les algorithmes déterminant les pages qui seront déplacées dans l'index de hachage dans MySQL . Il convient de noter que les erreurs d'index de hachage adaptatif ont un coût plus élevé que si l'index de hachage adaptatif n'était pas présent du tout.

Mémoire
innodb_file_per_table

- Il existe deux options pour stocker vos fichiers de données InnoDB. Vous pouvez toutes les stocker dans l'espace de table système (ibdata1) ou vous pouvez stocker chaque table dans son propre fichier. Notez que si vous avez activé fichier par table, il y aura 2 fichiers fichiers par table, un fichier frm contenant la définition et les métadonnées de la table et un fichier ibd contenant le contenu de la table.

Mémoire
innodb_file_per_table

- Par défaut: 1 (activé)
- Remarques sur le réglage: Il existe certaines considérations de performances pour le réglage de cette variable. Vous pouvez accélérer les opérations sur des instructions telles que ‘truncate table’ et contribuer à réduire les conflits de descripteurs de fichiers en répartissant les données sur plusieurs fichiers. Toutefois, l’avantage réel d’utiliser fichier par table est de faciliter la maintenance de la base de données. Vous pouvez facilement déplacer des espaces de tables et récupérer des espaces de tables où vous n'auriez pas pu autrement avant, lorsque tout était stocké dans l'espace de tables système.

Mémoire
innodb_file_format

- Actuellement, les deux formats de fichier pour InnoDB sont ‘antilope’ et ‘barracuda’. Barracuda est un nouveau format de fichier mis à la disposition de MySQL 5.1 dans le cadre du plug-in InnoDB. Il a été livré avec MySQL 5.5. Le format de fichier Barracuda prend en charge les formats de lignes «compacts» et «redondants» existants, mais comprend de nouveaux formats de lignes «dynamique» et «compressé».

Mémoire
innodb_file_format

- Par défaut: antilope (format)
- Notes de réglage: Il y a deux raisons principales pour passer à barracuda
- Vous souhaitez activer la compression de vos données afin de réduire le surcoût d'E / S au prix d'une utilisation accrue de la CPU. Si tel est le cas, vous voudrez utiliser barracuda avec le format de ligne compressé. Ou votre table contient des données hors page, telles que BLOB ou TEXT, qui sont trop volumineuses pour tenir sur la page, et le contenu complet des données est généralement accessible si elles doivent être lues. Dans ce cas, vous pouvez utiliser le format de ligne dynamique.
- Si vous utilisez le format de fichier antelope, ses deux formats de lignes (compact et redondant) stockeront les 768 premiers octets des données de type blob / text dans la page, puis établiront un pointeur indiquant où le reste des données peut se trouver. a trouvé. C'est bien si vous allez généralement rechercher ou chercher dans le contenu des 768 premiers octets, mais si ce n'est pas le cas, vous perdez de l'espace dans les pages qui pourrait être utilisé pour des données non blob / text. Le format de ligne dynamique ne stocke qu'un pointeur de 20 octets sur les données hors page. Utilisez-le donc si vous allez rarement accéder à vos données hors page ou si vous devez accéder à plus de 768 octets plus fréquemment.

Mémoire
innodb_open_files

- Une table InnoDB ouverte est un état logique dans lequel la table peut se trouver. Cette variable indique simplement combien de ces tables peuvent être dans cet état ouvert.
- Défaut:
  - Si innodb_file_per_table est désactivé = 300 (nombre de tables autorisées à être en statut ouvert)
  - Si innodb_file_per_table est activé = 300 ou la valeur de table_open_cache, la valeur la plus élevée

Mémoire
innodb_autoextend_increment

- Ceci définit la taille à laquelle l'espace de table système InnoDB (ibdata1) doit être étendu s'il manque d'espace. Cela s'applique uniquement à l'espace de table système et n'a pas d'incidence sur la croissance des fichiers de table créés par fichier InnoDB par table, pas plus que sur les autres espaces de table créés explicitement et qui se développent chacun de 4 Mo.
- Par défaut: 64 (Mo)
- Remarques sur le réglage: Si vous n'utilisez pas le fichier InnoDB par table et que vous vous attendez à une croissance rapide de votre tablespace système, envisagez d'augmenter cette valeur. Chaque fois que l'espace table doit être étendu, il utilise un mutex pendant l'extension du fichier, ce qui peut interrompre l'activité du moteur de stockage en cours.

Mémoire
innodb_page_size

- Définition: Ceci définit la taille de la page pour les données dans innodb. Ceci est défini une fois lorsque l'instance MySQL est créée et ne peut pas être modifié
- Par défaut: 16k
- Remarques sur le réglage: Lorsque vous configurez une instance MySQL, vous pouvez réduire la taille de la page InnoDB à celle de la taille de la page du disque lui-même afin d'éviter que les données non modifiées ne soient réécrites. Ceci est courant pour les SSD car ils utilisent généralement des blocs plus petits. Cependant, cela réduit la largeur de ligne maximale de votre enregistrement InnoDB et ne peut pas être modifié une fois l'instance créée, vous devrez utiliser une sauvegarde et une restauration logiques afin d'ajuster la taille de la page à une date ultérieure

Mémoire
innodb_checksum_algorithm

- Lorsque InnoDB écrit une page sur le disque, il calcule une somme de contrôle des données de la page et enregistre cette somme à la fin de la page. Lorsque la page est extraite du disque, InnoDB vérifie la somme de contrôle à la fin de la page afin de s'assurer que les données de la page correspondent à ce qui avait été écrit à l'origine. Cela permet de détecter des problèmes de stockage tels que la pourriture de bits. C'est également la base de l'outil innochecksum, qui recalcule la somme de contrôle pour chaque page et la compare à la valeur stockée.
- Cette variable vous permet de spécifier l'algorithme de somme de contrôle utilisé ou de spécifier qu'aucun algorithme ne doit être utilisé.
- Valeur par défaut: InnoDB
- Remarques sur le réglage: Bien que vous ayez la possibilité de désactiver le contrôle de contrôle et que cela améliorerait les performances, cela n'est pas recommander.

Mémoire
innodb_log_file_size

- La taille de chaque journal de reprise dans le groupe de journaux. Le journal de rétablissement stocke des informations de transaction sur les modifications apportées aux données InnoDB, mais qui n'ont pas encore été copiées du pool de mémoire tampon dans la table. Les pages du pool de mémoire tampon modifiées mais non encore vidées sur le disque sont considérées comme "sales". Ces pages modifiées peuvent rester dans le pool de mémoire tampon tant que la modification associée est stockée dans le journal de restauration. En utilisant les journaux redo, il n'est pas nécessaire de vider immédiatement les modifications sur la table sur le disque lors de la validation de la transaction, ce qui augmente les performances tout en permettant à InnoDB de rester compatible ACID.
- Par défaut: 50331648 (48 Mo)
- Remarques sur le réglage: Plus vous disposez d'espace de journalisation, moins InnoDB doit se vider de manière agressive. Assurez-vous que la combinaison de la taille du fichier journal et des fichiers journaux du groupe (taille du fichier journal * fichiers journaux du groupe) est suffisamment grande pour gérer au moins une heure de trafic.

innodb_log_files_in_group

- Cela indique combien de fichiers de journalisation seront présents dans le groupe de journaux. Ceci, en combinaison avec innodb_log_file_size (innodb_log_files_in_group * innodb_log_file_size) détermine la taille du journal de reprise.
- Par défaut: 2 (fichiers)
- Remarques sur le réglage: Cette variable a été ajoutée à l'origine pour permettre l'utilisation de systèmes de fichiers incapables de gérer des fichiers volumineux.
- Dans le cas où un journal de restauration volumineux était nécessaire, vous pouvez l'étendre sur plusieurs fichiers plus petits au lieu d'un petit nombre de fichiers plus volumineux. Le seul avantage que vous pourriez avoir à avoir plusieurs fichiers plus petits serait de réduire le risque de conflit entre le processus de vidage écrit dans le journal de reprise et les processus de purge qui suppriment les données du journal de reprise, mais ces deux processus fonctionnent en séquence. par conséquent, si tout fonctionne avec une grande efficacité, il est probable que ces processus fonctionneront avec le même fichier. Il est généralement considéré comme la meilleure pratique de laisser cette valeur à la valeur par défaut de 2.

Mémoire
innodb_log_buffer_size

- il s'agit de la mémoire tampon utilisée pour collecter les informations de transaction qui seront écrites dans le journal de rétablissement. Les données seront collectées ici en tant que transaction exécutée, puis seront écrites dans le journal de restauration à un intervalle désigné par la valeur de la variable Innodb_flush_log_at_trx_commit. Si vous avez une transaction importante qui écrit plus de modifications que le tampon de journal ne peut en contenir, le tampon devra être vidé dans le journal de reprise, éventuellement plusieurs fois, avant que la transaction ne soit validée pour ne pas dépasser sa capacité maximale.
- Par défaut: 8388608 (8 Mo)
- Remarques sur le réglage: assurez-vous qu'il est suffisamment volumineux pour éviter que les transactions volumineuses ne soient annulées plusieurs fois avant la validation. Il est conseillé de s'assurer que la quantité de données dans la mémoire tampon du journal est égale ou inférieure à 30% à la taille disponible de la mémoire tampon du journal.

IO
innodb_flush_log_at_trx_commit

- Cela détermine la méthode que InnoDB utilisera pour écrire / vider les données récemment modifiées dans le journal de rétablissement et vous permettra de réduire les propriétés de conformité ACID de la nouvelle connexion afin d'améliorer les performances. Il y a trois valeurs à choisir.
  - 1 = (performances les plus basses, conforme à l'ACID) Lorsqu'une transaction est validée, le contenu du tampon de journal est écrit et vidé dans le fichier journal redo sur le disque.
  - 2 = (meilleures performances, non conforme à ACID) Le contenu du tampon de journal est écrit dans le fichier journal de reprise après chaque validation de transaction, mais le fichier n'est vidé sur le disque que sur un intervalle de temps, en secondes, spécifié par la variable innodb_flush_log_at_timeout.
  - 0 = (performances optimales, non conforme à ACID) Le contenu du tampon de journal est écrit dans le journal de restauration sur un intervalle de temps, en secondes, spécifié par la variable innodb_flush_log_at_timeout, puis l'écriture est vidée sur le disque.
- Par défaut: 1
- Notes de réglage: Les valeurs 0 et 2 sont utilisées pour surmonter les pics DML inattendus, surmonter les retards de réplication temporaires .

IO
innodb_flush_log_at_timeout

- C'est le nombre de secondes qu'InnoDB attendra pour vider les modifications survenues dans le journal de reprise sur le disque si innodb_flush_log_at_trx_commit est définie sur une valeur autre que 1.
- Par défaut: 1 (seconde)
- Remarques sur le réglage: Si vous avez défini innodb_flush_log_at_trx_commit sur une valeur autre que 1, vous pouvez encore améliorer les performances en augmentant cette valeur. Toutefois, dans le cas d'une panne de MySQL, il s'agit du nombre de secondes de transactions InnoDB potentiellement perdues.

IO
sync_binlog

- Lorsque InnoDB termine une transaction, le journal binaire est mis à jour dans le cadre du processus de validation. La partie DML (insertions, mises à jour, suppressions) de la transaction est placée dans une mémoire tampon au niveau de la session appelée cache de transaction jusqu'à la validation. À ce stade, toutes les données de transaction sont formées dans des validations de groupe avec d'autres transactions de validation et sont écrites dans le journal binaire avant que la validation ne soit considérée comme terminée. Cette variable spécifie combien de validations de groupe de transactions peuvent être traitées avant l'appel de fdatasync () et le vidage des entrées du journal binaire en attente sur le disque.
- Par défaut: 0 (désactivé)
- Remarques sur l'optimisation: le seul moyen de garantir que votre journal binaire restera parfaitement synchronisé avec vos transactions InnoDB consiste à définir la valeur de cette variable sur 1.

IO
innodb_adaptive_flushing

- Innodb doit vider les pages modifié du pool de mémoire tampon vers l'espace table dans le cadre du processus de contrôle, ce qui vous assure de ne pas manquer d'espace de journalisation. Notez que le redo log est cyclique et que les informations de données doivent être purgées afin de laisser de la place pour les informations de transaction entrantes. La suppression d'entrées de l'espace de journalisation nécessite le vidage des pages modifiées dans l'espace table.
- Si le vidage adaptatif est activé, le taux de vidage de la page sale s'ajuste automatiquement en fonction de la charge de travail du système actuel. Vider trop de pages à la fois peut entraîner des conflits pour d'autres opérations, à la fois en lecture et en écriture. Si trop peu de pages sont effacées, InnoDB risque de prendre du retard dans son processus de vérification et risque de manquer d'espace de journalisation. Le rinçage adaptatif garantit le maintien de l'équilibre.
- Par défaut: 1 (ON)
- Notes de réglage: cette option doit être activé, sauf si vous souhaitez qu'InnoDB prenne en compte les caractéristiques des versions précédentes de MySQL, vous pouvez le désactiver.

IO
innodb_flush_neighbors

- Si cette variable est définie sur activé, lorsque InnoDB exécute le vidage d'une page du pool de mémoire tampon vers le stockage persistant, il vérifie si d'autres pages modifiées du pool de mémoire tampon appartiennent dans la même mesure et, le cas échéant, se vident. ceux-ci également afin de regrouper les écritures et d'éviter le temps de latence de la recherche sur disque. Les options pour cette variable sont les suivantes…
  - 1 = Activé, mais élimine uniquement les pages voisines modifiées du pool de mémoire tampon si elles ont la même étendue et si elles sont considérées comme contiguës.
  - 2 = Activé, mais l'exigence relative aux pages contiguës n'est pas présente dans ce paramètre, ce qui signifie que le vidage du voisin est plus susceptible de se produire.
  - 0 = désactivé
- Par défaut: 1
- Remarques sur le réglage: Désactivez lorsque vous utilisez des disques SSD ou une solution d'E / S rapide.

IO
innodb_flush_method

- Cela définit les appels système qu'InnoDB utilisera pour transférer les données vers un stockage physique pour les fichiers de données et les fichiers journaux.
- Par défaut: Null (sous Linux, fsync est la valeur par défaut, sous Windows, async_unbuffered par défaut)
- Remarques concernant le réglage: O_DIRECT devrait être pris en compte pour la plupart des charges de travail modernes, en particulier si vous disposez d’un pool de mémoire tampon InnoDB correctement configuré.

IO
innodb_doublewrite

- Le tampon de double écriture est utilisé pour s'assurer qu'une demande d'écriture d'une page sur le disque est réellement effectuée sur le disque avant que les enregistrements de modification associés ne soient purgés du journal de restauration InnoDB. Plus précisément, il est là pour se protéger contre les écritures de page partielles. Les pages sont écrites dans le tampon d'écriture double avant d'être écrites dans l'espace table. Lorsque InnoDB effectue une récupération, il vérifie l'intégrité de la page en comparant les données avec la somme de contrôle associée à la fin de la page. Cela fonctionnera pour les deux pages de la table et du tampon d'écriture double. S'il trouve une page dans l'espace table qui est incompatible avec sa somme de contrôle, il la récupérera à partir du tampon d'écriture double. S'il trouve une page dans la mémoire tampon de double écriture incompatible avec sa somme de contrôle, il le supprime
- Par défaut: On
- Remarques sur le réglage: Le tampon d'écriture double est vraiment petit, seulement assez grand pour supporter 100 pages, et est de nature séquentielle. Bien qu'il soit nécessaire d'écrire deux fois la page, la surcharge d'E / S n'est pas double, mais elle est notable.

IO
innodb_read_ahead_threshold

- Innodb a 2 méthodes différentes de lecture à l'avance et cette variable est directement liée à la méthode de lecture linéaire à l'avant. Ce paramètre indique le nombre de pages devant être lues de manière séquentielle dans une étendue (1 mégaoctet de données de page, généralement 64 pages sauf si vous utilisez le format de ligne de compression) pour que la lecture soit anticipée. Lorsque cela se produit, toutes les pages de l'extension suivante sont automatiquement extraites. Par exemple, si vous conservez la valeur par défaut de 56, lorsque InnoDB lit les données d'une extension si 56 pages sont insérées de manière séquentielle, le contenu complet de l'extension suivante est automatiquement extrait.
- Par défaut: 56 (pages)
- Remarques sur le réglage: La lecture anticipée a été implémentée à l'origine pour réduire le temps de recherche nécessaire pour extraire les informations du disque vers le pool de mémoire tampon. Si vous avez des disques SSD, vous pouvez envisager d'augmenter la valeur de cette variable pour rendre la lecture linéaire moins probable.

IO
innodb_read_io_threads / innodb_write_io_threads

- Innodb dispose d'une série de threads d'arrière-plan qui gèrent les requêtes d'E / S en arrière-plan et les sépare par des lectures et des écritures. Chaque thread peut gérer 256 demandes d'E / S en attente.
- Pour les lectures, il s'agirait d'un ordre de requête en lecture anticipée Par défaut: 4 ((chacun)
- Notes de réglage: Les threads gèrent désormais 256 requêtes en attente, ce qui signifie que l'augmentation de cette variable a moins d'impact que dans les versions précédentes de MySQL, mais uniquement si vous êtes utilisant les E / S asynchrones de linux (si vous êtes sur un système linux avec innodb_use_native_aio activé, valeur par défaut). Cela est dû au fait que les demandes d'E / S asynchrones sont transférées vers le système d'exploitation beaucoup plus rapidement car elles sont stockées dans des caches au niveau du système d'exploitation.

IO
innodb_io_capacity

- Il s'agit de la limite supérieure d'opérations d'E / S qu'InnoDB utilisera pour les tâches d'E / S en arrière-plan, telles que le vidage des pages non conformes et le déplacement des données d'index secondaire du tampon de modifications dans la table associée.
- Valeur par défaut: 200 (opérations d'E / S)
- Remarques sur le réglage: Cela devra être augmenté pour la majorité des serveurs sur lesquels MySQL est en cours d'installation. Particulièrement sur les systèmes avec E / S rapides ou SSD. Lors de la configuration, vous devez vérifier les spécifications du fabricant pour l'IOPS et l'utiliser comme base. Cependant, je ne recommanderais pas de définir cette valeur sur la valeur maximale. Vous devez plutôt prendre en compte 66% du nombre maximal d'IOPS disponibles pour le lecteur.

IO
innodb_io_capacity_max

- Lorsque InnoDB doit vider de manière plus agressive afin d'éviter de manquer d'espace de journalisation, il s'étend au-delà du paramètre de variable innodb_io_capacity, mais ne dépasse jamais celui défini par innodb_io_capacity_max. C'est sans doute la plus importante des deux variables à définir car elle définit la limite supérieure stricte et empêche InnoDB d'atteindre un point où elle atteindra la saturation lors du stockage physique.
- Valeur par défaut: 2000 (opérations d'E / S)
- Remarques sur le réglage: pour la plupart des systèmes, cette valeur doit être augmentée avec la valeur de la variable innodb_io_capacity.

IO
innodb_fast_shutdown

- Cela détermine ce que InnoDB va faire lorsque vous demandez à MySQL de s'arrêter.
  - 0 = arrêt lent. Cela exécutera toutes les étapes lors d'un arrêt rapide (paramètre variable: 1) ainsi que d'une purge complète de tous les enregistrements en attente de suppression sur le disque et d'une fusion complète du tampon de modification.
  - 1 = arrêt rapide. Cela ignorera la purge complète et modifiera la fusion du tampon. Mais les pages endommagées sont toujours vidées sur le disque et les tables sont contrôlées.
  - 2 = arrêt brutal. Arrêtez le moteur InnoDB comme si MySQL venait de tomber en panne. Aucune page n'est vidée sur le disque, aucune purge ne se produit, aucune fusion de tampon de modification ne se produit. Lors du démarrage, InnoDB devra procéder à une récupération et risquerait de perdre des données, voire de ne pas pouvoir démarrer du tout sans passer en mode de récupération forcée.
- Par défaut: 1
- Remarques sur le réglage: Laissez ce paramètre à 1. Définissez temporairement la valeur 0 avant d'arrêter MySQL pour une mise à niveau.

Consistence
foreign_key_checks

- Si activé, les clés étrangères sont observées. Si elles sont désactivées, les clés étrangères sont ignorées.
- Par défaut: 1 (ON)
- Remarques sur le réglage: vous pouvez désactiver cette fonction lorsque vous n'avez pas besoin d'observer les FK. Cela se produit généralement lorsque vous devez importer une grande quantité de données ou restaurer une sauvegarde logique lorsque vous savez que les données en cours de chargement sont cohérentes et ne nécessitent pas de validation lors de l'insertion d'un enregistrement. Cela peut être désactivé au niveau de la session. Si vous envisagez de supprimer définitivement cette fonctionnalité afin de réduire les frais généraux liés aux clés étrangères, vous pouvez envisager de supprimer les clés étrangères

Consistence
innodb_stats_persistent

- InnoDB détermine son plan d'exécution de la requête en fonction des statistiques collectées pour les index faisant partie de la ou des table (s) interrogée (s). Cette variable vous permet de stocker ces statistiques de manière persistante (trouvée dans les tables mysql.innodb_table_stats et mysql.innodb_index_stats) afin que vous ne soyez pas obligé de recalculer les statistiques lors du premier accès au tableau.
- Par défaut: ON
- Remarques sur le réglage: L'écriture de ces informations sur le disque est minime et peut vous éviter des contrôles d'index inutiles après un redémarrage de MySQL.

Consistence
innodb_stats_auto_recalc

- cette variable vous permet d'activer ou de désactiver une fonctionnalité qui recalcule les statistiques d'index et met à jour les tables mysql.innodb_table_stats et mysql.innodb_index_stats lorsque 10% des données de la table ont été modifiées.
- Par défaut: ON

Concurrence
innodb_thread_concurrency

- Cette variable spécifie le nombre de threads pouvant être utilisés par InnoDB lors de son interaction avec le système d'exploitation et limite ainsi le nombre de threads MySQL ayant accès au moteur InnoDB à un moment donné. Cela a été mis en place afin de limiter le nombre de threads, donc d'essayer de réduire le nombre de changements de contexte système dans MySQL. Cependant, sur les nouveaux matériels, la commutation de contexte matériel est moins coûteuse. Par conséquent, la valeur par défaut est de ne pas limiter le nombre de threads disponibles.
- Par défaut: 0 (threads) (aucune limite sur les threads générés)
- Remarques sur le réglage: avec les systèmes modernes et les versions de MySQL, il est préférable de laisser cette variable à sa valeur par défaut, mais vous pouvez également envisager de la modifier si vous hébergez MySQL sur un serveur hébergeant également d'autres processus.

Concurrence
innodb_thread_sleep_delay

- Si le nombre de threads en cours d'exécution correspond à la valeur de innodb_thread_concurrency (en supposant que sa valeur n'est pas 0), une nouvelle opération / requête demandant un thread se verra refuser l'accès au noyau InnoDB, puis attendra un nombre de microsecondes égal à la valeur de cette variable avant d'essayer une fois de plus de compléter sa demande du moteur de stockage. S'il ne parvient toujours pas à le faire après cette nouvelle tentative, il est placé dans la file d'attente des opérations en attente d'un thread, ce qui déclenche un changement de contexte. En effectuant une vérification avant d'entrer dans la file d'attente, cela réduit les risques de changement de contexte au niveau du système.
- Par défaut: 10000 (microsecondes) (0,01 seconde)
- Notes de réglage: Vous ne devez ajuster cette variable que si innodb_thread_concurrency a une valeur autre que 0.

Concurrence
Innodb_autoinc_lock_mode

- Lorsque de nouveaux enregistrements sont insérés dans une table avec une colonne auto_increment, un verrou est mis en place pour que l'ordre des insertions puisse être répété lors de la lecture via le journal binaire ou via la réplication. Pour plus d'informations sur ce mécanisme de verrouillage, reportez-vous au guide de référence MySQL. Les valeurs acceptables sont 0, 1 et 2.
- Par défaut: 1 (mode de verrouillage consécutif)
- Si vous avez un seul serveur sans esclaves pour lequel la cohérence des journaux binaire n'est pas requise pour la récupération à un moment précis, définissez le mode de verrouillage sur 2

Connexions, droits d'accès, sécurité
Gestion des utilisateurs et de leurs privilèges

- Privilège des utilisateurs
- Commande GRANT et REVOKE
- Mise en place des rôles

Gestion des droits et privilèges

- MySQL implémente un système sophistiqué de contrôle d'accès et de privilèges vous permettant de créer des règles d'accès complètes pour la gestion des opérations client et d'empêcher efficacement les clients non autorisés d'accéder au système de base de données.

Type de privilège sous MySQL

La fonction principale du système de privilèges MySQL est d'authentifier un utilisateur qui se connecte à partir d'un hôte donné et de lui associer des privilèges sur une base de données telle que SELECT, INSERT, UPDATE et DELETE.

Il y a des choses que vous ne pouvez pas faire avec le système de privilège MySQL:

- Vous ne pouvez pas spécifier explicitement qu'un utilisateur donné doit se voir refuser l'accès. Autrement dit, vous ne pouvez pas explicitement faire correspondre un utilisateur et ensuite refuser la connexion.
- Vous ne pouvez pas spécifier qu'un utilisateur dispose des privilèges pour créer ou supprimer des tables dans une base de données, mais pas pour créer ou supprimer la base de données elle-même. Un mot de passe s'applique globalement à un compte.
- Vous ne pouvez pas associer un mot de passe à un objet spécifique tel qu'une base de données, une table ou une routine.

Schema Mysql

- La base de données mysql contient cinq tables de privilèges. Vous manipulez ces tables indirectement via des instructions telles que GRANT et REVOKE.
  - user: contient les colonnes compte d'utilisateur et privilèges globaux. MySQL utilise la table user pour accepter ou refuser une connexion depuis un hôte. Un privilège accordé dans la table user est effectif pour toutes les bases de données sur le serveur MySQL.
  - db: contient les privilèges de niveau base de données. MySQL utilise la table db pour déterminer la base de données à laquelle un utilisateur peut accéder et à partir de quel hôte. Un privilège accordé au niveau de la base de données dans la table de base de données s'applique à la base de données et tous les objets appartiennent à cette base de données, par exemple des tables, des déclencheurs, des vues, des procédures stockées, etc.

Schema Mysql

- table_priv et columns_priv: contient les privilèges au niveau table et au niveau colonne. Un privilège accordé dans la table table_priv s'applique à la table et à ses colonnes, tandis qu'un privilège accordé à la table columns_priv s'applique uniquement à une colonne spécifique d'une table.
- procs_priv: contient les privilèges de fonctions et de procédures stockées

Type de privilège sous MySQL

Les privilèges accordés à un compte MySQL déterminent les opérations que le compte peut effectuer. Les privilèges MySQL diffèrent selon le contexte d'application et le niveau de fonctionnement:

- administratifs permettent aux utilisateurs de gérer le fonctionnement du serveur MySQL. Ces privilèges sont globaux car ils ne sont pas spécifiques à une base de données particulière.
- base de données s'appliquent à une base de données et à tous les objets qu'elle contient. Ces privilèges peuvent être accordés pour des bases de données spécifiques ou globalement, de sorte qu'ils s'appliquent à toutes les bases de données.
- pour des objets de base de données tels que des tables, des index, des vues et des routines stockées peuvent être accordés pour des objets spécifiques dans une base de données

Privileges

- ALL, ALL PRIVILEGES:Ces spécificateurs de privilège sont des raccourcis pour «tous les privilèges disponibles à un niveau de privilège donné»
- ALTER Permet d'utiliser l'instruction ALTER TABLE pour modifier la structure des tables. ALTER TABLE requiert également les privilèges CREATE et INSERT. Renommer une table nécessite ALTER et DROP sur l'ancienne table, CREATE et INSERT sur la nouvelle table.
- ALTER ROUTINE Permet l'utilisation d'instructions qui modifient ou suppriment les routines stockées
- CREATE Permet l'utilisation d'instructions qui créent de nouvelles bases de données et tables.

Privilèges

- CREATE ROLE. Active l'utilisation de l'instruction CREATE ROLE. (Le privilège CREATE USER permet également d'utiliser l'instruction CREATE ROLE.)
- CREATE ROUTINE Permet l'utilisation d'instructions qui créent des routines stockées (procédures stockées et fonctions). table, telle que DROP TABLE, INSERT, UPDATE ou SELECT.
- CREATE USER Active l'utilisation des instructions ALTER USER, CREATE ROLE, CREATE USER, DROP ROLE, DROP USER, RENAME USER et REVOKE ALL PRIVILEGES.
- CREATE VIEW Active l'utilisation de l'instruction CREATE VIEW. DELETE Permet aux lignes d'être supprimées des tables d'une base de données.
- DROP Permet l'utilisation d'instructions qui suppriment (suppriment) les bases de données, les tables et les vues existantes. Le privilège DROP est requis pour utiliser l'instruction ALTER TABLE ...

Privilèges

- DROP ROLE. (Le privilège CREATE USER permet également d'utiliser l'instruction DROP ROLE.) Vous permet d'accorder ou de révoquer à d'autres utilisateurs les privilèges que vous possédez vous-même.
- INDEX Permet l'utilisation d'instructions qui créent ou suppriment (suppriment) des index. INDEX s'applique aux tables existantes. Si vous disposez du privilège CREATE pour une table, vous pouvez inclure des définitions d'index dans l'instruction CREATE TABLE.
- INSERT Permet aux lignes d'être insérées dans les tables d'une base de données. INSERT est également requis pour les instructions table-maintenance ANALYZE TABLE, OPTIMIZE TABLE et REPAIR TABLE.
- LOCK TABLES Permet l'utilisation d'instructions explicites LOCK TABLES pour verrouiller les tables pour lesquelles vous disposez du privilège SELECT. Cela inclut l'utilisation de verrous en écriture, ce qui empêche d'autres sessions de lire la table verrouillée.

Privilèges

- SELECT Permet aux lignes d'être sélectionnées à partir des tables d'une base de données.
- SHUTDOWN Active l'utilisation des instructions SHUTDOWN et RESTART,
- TRIGGER Active les opérations de déclenchement. Vous devez disposer de ce privilège pour qu'une table crée, supprime, exécute ou affiche des déclencheurs pour cette table.
- UPDATE Permet aux lignes d'être mises à jour dans les tables d'une base de données.

Privilèges

	Serveur	Base de Données	Table	Colonne	Procedure
ALL					
ALTER	■	■	■		
ALTER ROUTINE	■	■			■
CREATE	■	■	■		
CREATE ROUTINE	■	■			
CREATE USER	■				
CREATE VIEW	■	■	■		
DELETE	■	■	■		
DROP	■	■	■		
EXECUTE	■	■	■		
GRANT OPTION	■	■	■		■
INDEX	■	■	■		
INSERT	■	■	■	■	
SELECT	■	■	■	■	
SHUTDOWN	■				
TRIGGER	■	■	■		
UPDATE	■	■	■	■	

GRANT

Après avoir créé un nouveau compte d'utilisateur, l'utilisateur ne dispose d'aucun privilège. Pour accorder des privilèges à un compte d'utilisateur, utilisez l'instruction GRANT. Voici une illustration de la syntaxe de l'instruction GRANT:

```
GRANT privilege,[privilege],.. ON privilege_level
TO user [IDENTIFIED BY password]
[REQUIRE tsl_option]
[WITH [GRANT_OPTION | resource_option]];
```

GRANT

- Tout d'abord, spécifiez un ou plusieurs privilèges après le mot clé GRANT. Si vous accordez plusieurs privilèges à l'utilisateur, chaque privilège est séparé par une virgule.
- Ensuite, spécifiez le niveau de privilège qui détermine le niveau auquel les privilèges s'appliquent. MySQL supporte les niveaux global (*. *), Base de données (database. *), Table (database.table) et colonne. Si vous utilisez le niveau de privilège de la colonne, vous devez spécifier une colonne ou une liste de colonnes séparées par des virgules après chaque privilège.
- Ensuite, placez l'utilisateur que vous souhaitez accorder des privilèges. Si l'utilisateur existe déjà, l'instruction GRANT modifie ses privilèges.

GRANT

Ensuite, vous indiquez si l'utilisateur doit se connecter au serveur de base de données via une connexion sécurisée telle que SSL, X059, etc.

Enfin, la clause facultative WITH GRANT OPTION vous permet d'accorder aux autres utilisateurs ou de supprimer d'autres utilisateurs les privilèges que vous possédez.

Pour accorder tous les privilèges au compte d'utilisateur super @ localhost, utilisez l'instruction suivante

- GRANT ALL ON *.* TO 'super'@'localhost' WITH GRANT OPTION

La clause ON *. * Désigne toutes les bases de données et tous les objets des bases de données. WITH GRANT OPTION permet à super @ localhost d'accorder des privilèges à d'autres utilisateurs.

GRANT

Pour créer un utilisateur disposant de tous les privilèges dans la base de données exemple classicmodels, utilisez les instructions suivantes:

- CREATE USER auditor @ localhost IDENTIFIED BY 'baleine' REQUIRE SSL;
- GRANT ALL on classicmodels. * TO auditor @ localhost;

Vous pouvez accorder plusieurs privilèges dans une seule instruction GRANT. Par exemple, vous pouvez créer un utilisateur capable d'exécuter les instructions SELECT, INSERT et UPDATE sur la base de données classicmodels à l'aide des instructions suivantes:

- CREATE USER rfc IDENTIFIED BY 'shark';
- GRANT SELECT, UPDATE, DELETE ON classicmodels. * TO rfc;

REVOKE

Afin de révoquer les privilèges d'un compte utilisateur, vous utilisez l'instruction MySQL REVOKE. MySQL vous permet de révoquer un ou plusieurs privilèges ou tous les privilèges d'un utilisateur.

Ce qui suit illustre la syntaxe de révocation de privilèges spécifiques d'un utilisateur:

REVOKE privilege_type [(column_list)] [, priv_type [(column_list)]] ... ON [type d'objet] privilege_level DE l'utilisateur [, utilisateur] ...

- Tout d'abord, spécifiez une liste de privilèges que vous souhaitez révoquer d'un utilisateur juste après le mot clé REVOKE. Vous devez séparer les privilèges par des virgules.
- Deuxièmement, spécifiez le niveau de privilège auquel les privilèges sont révoqués dans la clause ON.
- Troisièmement, spécifiez le compte d'utilisateur pour lequel vous souhaitez révoquer les privilèges dans la clause FROM.

Notez que pour révoquer les privilèges d'un compte d'utilisateur, vous devez disposer du privilège GRANT OPTION et des privilèges que vous révoquez.

Roles MySql

Pour faciliter les choses, MySQL a fourni un objet appelé rôle, qui est une collection nommée de privilèges.

Si vous souhaitez accorder le même ensemble de privilèges à plusieurs utilisateurs, procédez comme suit:

- Tout d'abord, créez un nouveau rôle.
- Deuxièmement, accordez des privilèges au rôle.
- Troisièmement, accordez le rôle aux utilisateurs. Si vous souhaitez modifier les privilèges des utilisateurs, vous devez modifier uniquement les privilèges du rôle attribué.

Les modifications prendront effet pour tous les utilisateurs auxquels le rôle a été attribué.

Create Role

Supposons que vous développiez une application utilisant unebase de données CRM.

Pour interagir avec la base de données CRM, vous devez créer des comptes pour les développeurs ayant besoin d'un accès complet à la base de données.

En outre, vous devez créer des comptes pour les utilisateurs n'ayant besoin que d'un accès en lecture et les autres utilisateurs ayant besoin d'un accès en lecture / écriture.

Pour éviter d'accorder des privilèges à chaque compte d'utilisateur individuellement, vous créez un ensemble de rôles et accordez les rôles appropriés à chaque compte d'utilisateur.
Pour créer de nouveaux rôles, utilisez l'instruction CREATE ROLE:

- CREATE ROLE crm_dev, crm_read, crm_write;

Grant Role

Pour accorder des privilèges à un rôle, vous utilisez l'instruction GRANT.

L'instruction suivante accorde tous les privilèges au rôle crm_dev:

- GRANT ALL ON crm. * TO crm_dev;

L'instruction suivante accorde le privilège SELECT au rôle crm_read: 1

- GRANT SELECT ON crm. * TO crm_read;

L'instruction suivante accorde les privilèges INSERT, UPDATE et DELETE au rôle crm_write:

- GRANT INSERT, UPDATE, DELETE ON crm. * TO crm_write;

Associer des utilisateurs aux rôles

Supposons que vous ayez besoin d'un compte utilisateur en tant que développeur, d'un compte utilisateur pouvant disposer d'un accès en lecture seule et de deux comptes utilisateur pouvant disposer d'un accès en lecture / écriture. Pour créer de nouveaux utilisateurs, utilisez les instructions CREATE USER comme suit:

- CREATE USER crm_dev1 @ localhost IDENTIFIED BY 'Secure $ 1782';

utilisateur d'accès en lecture

- CREATE USER crm_read1 @ localhost IDENTIFIED BY 'Secure $ 5432';

utilisateurs en lecture / écriture

- CREATE USER crm_write1 @ localhost IDENTIFIED BY 'Secure $ 9075';
- CREATE USER crm_write2 @ localhost IDENTIFIED BY 'Secure $ 3452';

Pour attribuer des rôles aux utilisateurs, vous utilisez l'instruction GRANT:

- GRANT crm_dev TO crm_dev1 @ localhost;
- GRANT crm_read TO crm_read1 @ localhost;
- GRANT crm_read, crm_write TO crm_write1 @ localhost, crm_write2 @ localhost;

Associer des rôles par défaut

Maintenant, si vous vous connectez à MySQL à l'aide du compte utilisateur crm_read1 et essayez d'accéder à la base de données CRM:

- mysql -u crm_read1 -p
- Entrer le mot de passe: **********
- mysql> USE crm;

L'instruction a émis le message d'erreur suivant:

- ERREUR 1044 (42000): Accès refusé pour l'utilisateur 'crm_read1' @ 'localhost' à la base de données 'crm'

En effet, lorsque vous attribuez des rôles à un compte d'utilisateur, les rôles ne deviennent pas automatiquement actifs lorsque le compte d'utilisateur se connecte au serveur de base de données.

Rôle par défaut

Si vous appelez la fonction CURRENT_ROLE ():

- SELECT current_role ();
- | NONE|

NONE est renvoyé, c'est-à-dire aucun rôle actif.

Pour spécifier les rôles qui doivent être actifs chaque fois qu'un compte utilisateur se connecte au serveur de base de données, vous utilisez l'instruction SET DEFAULT ROLE.

L'instruction suivante définit la valeur par défaut pour le compte crm_read1 @ localhost avec tous ses rôles attribués.

- SET DEFAULT ROLE ALL TO crm_read1 @ localhost;

Rôle Actif

Un compte utilisateur peut modifier les privilèges effectifs de l'utilisateur actuel dans la session en cours en spécifiant les rôles attribués actifs.

- SET ROLE NONE L'instruction suivante définit le rôle actif sur NONE, ce qui signifie qu'aucun rôle actif.
- SET ROLE ALL Pour définir les rôles actifs sur tous les rôles attribués, vous utilisez
- SET DEFAULT ROLE Pour définir les rôles actifs sur les rôles par défaut définis par l'instruction
- Pour définir des rôles nommés actifs, vous utilisez: 1 SET ROLE grant_role_1, grant_role_2, ...

Connexions, droits d'accès, sécurité
Type de droits

- Quel sont les types de droits?
- Quand les droits prennent effet?

Droits d'administration

Les droits d'administration permettent d'administrer le serveur MySQL. Il doivent en principe être réservés aux comptes administrateur.

Ils sont définis au niveau global, cad à tous les objet du serveur.

Droit Administration

Droit	Description
CREATE TEMPORARY TABLES	Crée de tables temporaires
CREATE USER	Créer, modifier, supprimer les comptes utilisateurs (CREATE, DROP, RENAME)
FILE	Lire ou écire dans les fichiers stockés sur la machine hôte du serveur MySQL(OUTFILE, LOAD DATA)
GRANT OPTION	Permet de transmettre ses droits à d'autres comptes
LOCK TABLES	Verrouiller les tables
PROCESS	Afficher les threads
RELOAD	Réinitialiser les journaux, les tables, les statistiques (FLUSH,RESET)
REPLICATION CLIENT	Superviser et gérer la réplication (SHOW MASTER STATUS, SHOW SLAVE STATUS)

Droit Administration

Droit	Description
REPLICATION SLAVE	Utilisé par le compte de réplication pour récupéré les évènements du journal binaire du maître
SHUTDOWN	Arrreter le serveur
SUPER	Executer diverses commandes d'administration (CHANGE MASTER TO, KILL, SET GLOBAL..)

Droits au niveau des schémas

Il est parfois nécessaire de définir des droits sur la globalité d'un schéma. Les droits portent alors sur l'ensemble des objets appartenant à ce schéma.

Droit Schémas

Droit	Description
ALTER	Modifier les schémas et les tables (ALTER SCHEMA/DATABASE/TABLE)
CREATE	Créer des schémas et des tables (CREATE SCHEMA/DATABASE/TABLE)
CREATE TEMPORARY TABLE	Créer des tables temporaires (CREATE TEMPORARY TABLES)
CREATE VIEW	Créer des vues (CREATE VIEW)
DELETE	Effeacer des enregistrements d'une table (DELETE)
DROP	Supprimer des schémas ou des tables (DROP SCHEMA/DATABASE/TABLE)
EVENT	Programmer les évènements de l'ordonnancer d'évènements (CREATE EVENT)
GRNT OPTION	Permet de transmettre ses droits à d'autres comptes

Droit Schémas

Droit	Description
INDEX	Créer et supprimer des index (CREATE/DROP/INDEX)
INSERT	Insérer des enregistrements dans une table (INSERT)
LOCK TABLES	Verrouiller les tables (LOCK TABLES)
SELECT	Afficher les enregistrements d'une tables (SELECT,DESCRIBE,SHOW,CREATE TABLE)
SHOW VIEW	Voir le code SQL d'une vue (SHOW CREATE VIEW)
UPDATE	Modifier des enregistrements (UPDATE)

Droits au niveau des tables

Il peut arriver que l'application ne fasse qu'insérer des données dans une table. Dans ce case, il est possible de spécifier des droits pour ne permettre de manipuler que la table et ses données.

Droits au niveau des tables

Droit	Description
ALTER	Modifier les tables (ALTER TABLE)
CREATE	Créer des tables (CREATE TALBE)
DELETE	Effacer des enregistrments d'une table (DELETE)
DROP	Supprimer des tables (DROP TABLE)
GRANT OPTION	Permet de transmettre ses droits à d'autres comptes
INDEX	Créer et supprimer des index (CREATE/DROP INDEX)
INSERT	Insérer des enregistrements dans une table
SELECT	Afficher les enregistrements d'une table
TRIGGER	Créer et supprimer des triggers (CREATE/DROP TRIGGER)
UPDATE	Modifier des enregistrements

Droits au niveau des colonnes

LA granularité la plus fine donne la possibilité d'ajouter, d'afficher et de modifier les données de certaines colonnes d'une table.

Les droits sur les colonnes empêchent les requêtes d'être mise dans le cache de requêtes.

Il n'est pas possible d'avoir le droit DELETE pour une colonne parce que la granularité d'un effacement est l'enregistrement.

Droits au niveau des colonnes

Droit	Description
INSERT	Insérer des enregistrements dans une table
SELECT	Affichier les enregistrement d'une table
UPDATE	Mdoifier des enregistrements

Droits sur les routines

LE système de droits s'applique également pour contrôler si un utilisateur peut créer, modifier ou supprimer des routines stockées

Droit	Description
CREATE ROUTINE	Créer des procédures et des fonctions stockées
ALTER ROUTINE	Modifier ou supprimer des procédures et des fonctions stockées
EXECUTE	Exécuter des procédures et des fonctions stockées
GRANT OPTION	Permet de transmettre ses droits à d'autre comptes.

Sécurisation des vues et des routines

L'accès aux objets de la base de données est sécurisé au niveau des comptes utilisateurs avec un système qui vous y autorise ou non.

Cependant, même si vous avez le droit pour accéder à l'objet, il n'est pas certain que vous ayez le droit d'accéder au données auxquelles il se réfère.

Pour exécuter une vue, il faut que l'utilisateur ait le droit de manipuler les objets de la vue.

Pour une vue, c'est la clause SQL SECURITY qui permet de paramétrer le comportement des objets sous jacent de la vue.

SQL SECURITY

- INVOKER: la vue est exécutée avec les droits de l'utilisateur
- DEFINER: la vue est exécutée avec les droits de son créateur
    - CREATE SQL SECURITY DEFINER VIEW <view name> as select…

- LA clause SQL SECURITY existe également pour les routines, son comportement est similaire.

FLUSH PRIVILEGES

Si vous modifiez directement les tables d'attribution à l'aide d'instructions telles que INSERT, UPDATE ou DELETE, vos modifications n'auront aucune incidence sur la vérification des privilèges jusqu'à ce que vous redémarriez le serveur ou que vous lui indiquiez de recharger les tables. Si vous modifiez directement les tables de droits sans oublier de les recharger, vos modifications sont sans effet tant que vous n'avez pas redémarré le serveur. Cela peut vous amener à vous demander pourquoi vos modifications ne semblent faire aucune différence!

Pour que le serveur recharge les tables d'attribution, effectuez une opération de vidage des privilèges. Cela peut être fait en émettant une instruction FLUSH PRIVILEGES ou en exécutant une commande mysqladmin flush-privileges ou mysqladmin reload.

Si vous modifiez les tables d'attribution indirectement à l'aide d'instructions de gestion de compte telles que GRANT, REVOKE, SET PASSWORD ou RENAME USER, le serveur remarque ces modifications et charge immédiatement les tables d'attribution en mémoire.

Connexions, droits d'accès, sécurité
Utilisation de SSL.

- Mise en place de SSL dans MySQL
- Associer un utilisateur a une connectivité SSL

Chiffrement des informations

- Les informations qui transitent entre un serveur et ses clients circulent en clair et peuvent être facilement interceptées.
- Pour contrer cela, il est possible de chiffrer la communication.
- Il existe plusieurs manières de créer un certificats. Deux utilitaires de création de certificat sont possibles, mysql_ssl_rsa_setup issu des outils MySQL et OpenSSL qui est une alternative du système.

mysql_ssl_rsa_setup

- L'utilitaire mysql_ssl_rsa_setup est un outil client de MySQL qui permet de générer automatiquement toutes les clefs nécéssaires à la mise en œuvre d'une connexion chiffrée.
- Shell> mysql_ssl_rsa_setup --datadir=/var/lib/mysql
- Le résultat est ainsi :

```
Generating a 2048 bit RSA private key
....+++
.......+++
writing new private key to 'ca-key.pem'
-----
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
.......+++
writing new private key to 'server-key.pem'
-----
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
....+++
..+++
writing new private key to 'client-key.pem'
-----
```

mysql_ssl_rsa_setup

- Les clefs sont ensuite bien créées dans le repertoire de destination (ici sous windows)

```
Directory of D:\mysql-5.7.10-winx64\data

01/13/2016  04:19 PM    <DIR>          .
01/13/2016  04:19 PM    <DIR>         ..
01/13/2016  04:16 PM                56 auto.cnf
01/13/2016  04:19 PM             1,679 ca-key.pem
01/13/2016  04:19 PM             1,074 ca.pem
01/13/2016  04:19 PM             1,078 client-cert.pem
01/13/2016  04:19 PM             1,675 client-key.pem
...
01/13/2016  04:19 PM             1,675 private_key.pem
01/13/2016  04:19 PM               451 public_key.pem
01/13/2016  04:19 PM             1,078 server-cert.pem
01/13/2016  04:19 PM             1,679 server-key.pem
```

OpenSSL

- L'utilitaire musql_ssl_rsa_setup vu précédemment générer des certificats auto signés. Une possibilité consiste à utiliser OpenSSL, en générant une autorité de certification.

```
# Créer un certificat de CA
openssl genrsa 2048 > ca-key.pem
openssl req -new -x509 -nodes -days 3600 \
        -key ca-key.pem -out ca.pem

# Créer un certificat de serveur, supprimer le mot de passe  et le signer
# server-cert.pem = clé publique, server-key.pem = clé privée
openssl req -newkey rsa:2048 -days 3600 \
        -nodes -keyout server-key.pem -out server-req.pem
openssl rsa -in server-key.pem -out server-key.pem
openssl x509 -req -in server-req.pem -days 3600 \
        -CA ca.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem
```

OpenSSL

- Puis vient la génération de la clef du client

```
#Créer un certificat client, supprimer le mot de passe  et le
signer
# client-cert.pem = public key, client-key.pem = private key


openssl req -newkey rsa:2048 -days 3600 \
        -nodes -keyout client-key.pem -out client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -req -in client-req.pem -days 3600 \
        -CA ca.pem -CAkey ca-key.pem -set_serial 01 -out
client-cert.pem
```

Configuration du mode SSL

Une fois les certificats générés, le serveur doit prendre en compte les clés générées et permettre aux utilisateurs de chiffrer leur connexion.

Les étapes de configuration du serveur sont ainsi:

- Création d'un utilisateur utilisateur SSL.
- Configuration du serveur
- Configuration du client

Création d'un utilisateur SSL

- Un compte utilisateur doit être utilisé pour se connecter en SSL.

CREATE USER "neoflow"@"%" IDENTIFIED BY "motdepasse" REQUIRE SSL;

- Si l'utilisateur existe déjà le contenu de la table mysql.user indique si oui ou non il peut se connecter en SSL

Select user,Host,ssl_type from mysql.user where User='neoflow'

- Et lui rajouter les droits

Alter user 'newflow'@'%' require SSL

- Enfin forcer la modification via

FLUSH PRIVILEGES

Configuration SSL du serveur

- Par défaut, le serveur MySQL autorise les connexions SSL(option –ssl).
- Ce dernier doit prendre en comtpe les certificats générés dans le fichier my.cnf et dans la section mysqld

```
[mysqld]
ssl-ca=ca.pem
ssl-cert=server-cert.pem
ssl-key=server-key.pem
```

Configuration du client

- Il faut en premier lieu copier du serveur vers le client le certificats, et les clefs clients (ca.pem, client-cert.pem, client-key.pem)
- Puis soit modifier le fichier my.cnf du client soit passer les paramètres au client mysql

```
mysql --ssl-ca=ca.pem \
      --ssl-cert=client-cert.pem \
      --ssl-key=client-key.pem
```

Mode SSL

Par défaut, les programmes clients MySQL tentent d'établir une connexion chiffrée si le serveur prend en charge les connexions chiffrées, avec un contrôle supplémentaire disponible via l'option --ssl-mode:

En l'absence d'une option --ssl-mode, les clients tentent de se connecter à l'aide d'un cryptage, retombant sur une connexion non cryptée si une connexion cryptée ne peut pas être établie. C'est également le comportement avec une option explicite --ssl-mode = PREFFERED.

- Avec --ssl-mode = REQUIRED, les clients nécessitent une connexion chiffrée et échouent s'il est impossible de l'établir.
- Avec --ssl-mode = DISABLED, les clients utilisent une connexion non chiffrée.
- Avec --ssl-mode = VERIFY_CA ou --ssl-mode = VERIFY_IDENTITY, les clients exigent une connexion chiffrée et effectuent également une vérification du certificat de l'autorité de certification du serveur et (avec VERIFY_IDENTITY) du nom d'hôte du serveur dans son certificat.

Moteurs de stockage et plug-ins

- Quelques plugin MySQL.
- Présentation du plugin rewriter

Architecture MySQL

- L’architecture de MySQL est une architecture modulaire à base de plugin
- L'API de plug-in permet la création de plug-ins qui implémentent plusieurs fonctionnalités:
  - FullText
  - Storage
  - Authentification
  - Password
  - Demons ..

INSTALL PLUGIN

- INSTALL PLUGIN plugin_name SONAME 'shared_library_name’
- Cette déclaration installe un plugin dans le serveur. Il nécessite le privilège INSERT pour la table système mysql.plugin.

- plugin_name est le nom du plugin tel que défini dans la structure de descripteur de plugin contenue dans la bibliothèque

Plugin Storage Engine

L’architecture de moteur de stockage enfichable utilisée par MySQL Server permet aux moteurs de stockage d’être écrits en tant que plug-ins, puis chargés et déchargés sur un serveur en cours d’exécution. Pour une description de cette architecture

Disponible en nombre sur https://mariadb.com/kb/en/library/storage-engines/

Full-Text Plugins

- MySQL dispose d'un analyseur intégré qu'il utilise par défaut pour les opérations de texte Full Text (analyse du texte à indexer ou analyse d'une chaîne de requête pour déterminer les termes à utiliser pour une recherche). L'analyseur de texte intégral intégré est pris en charge avec les tables InnoDB et MyISAM.
- L'API du plugin vous permet d'utiliser un analyseur de texte intégral autre que l'analyseur de texte intégral intégré par défaut.
  - CREATE TABLE t
  - (
  - doc CHAR(255),
  - FULLTEXT INDEX (doc) WITH PARSER parser_name
  -) ENGINE=InnoDB;

Daemon Plugins

- Un plugin daemon est un type simple de plugin utilisé pour le code devant être exécuté par le serveur mais ne communiquant pas avec lui. Les distributions MySQL incluent un exemple de plug-in daemon qui écrit des messages de pulsation périodiques dans un fichier.

Plugin d’Authentification

- Des plug-ins d'authentification existent à la fois du côté serveur et du côté client.
- Les plug-ins côté serveur implémentent des méthodes d'authentification à utiliser par les clients lorsqu'ils se connectent au serveur. Un plug-in côté client communique avec un plug-in côté serveur pour fournir les informations d'authentification requises.
- Un plug-in côté client peut interagir avec l'utilisateur en effectuant des tâches telles que la sollicitation d'un mot de passe ou d'autres informations d'authentification à envoyer au serveur.

Plugin de Password

- Le serveur MySQL fournit une interface pour écrire des plugins testant les mots de passe. Un tel plugin implémente deux fonctionnalités:

- Rejet des mots de passe trop faibles dans les instructions attribuant des mots de passe (telles que les instructions CREATE USER et ALTER USER).

- Évaluation de la force des mots de passe potentiels pour la fonction SQL VALIDATE_PASSWORD_STRENGTH ().

Plugin Rewriter

- MySQL supporte les plugins de réécriture de requêtes qui peuvent examiner et éventuellement modifier les instructions SQL reçues par le serveur avant que le serveur ne les exécute.
- Un plugin côté serveur nommé Rewriter examine les instructions et peut les réécrire, en fonction de son cache en mémoire et de règles de réécriture.

- Ces déclarations sont sujettes à la réécriture:
  - Depuis MySQL 8.0.12: SELECT, INSERT, REPLACE, UPDATE et DELETE.
  - Avant MySQL 8.0.12: SELECT uniquement.

Installation de Plugin

- Pour installer ou désinstaller le plugin Rewriter query rewrite, choisissez le script approprié situé dans le répertoire de partage de votre installation MySQL:

- install_rewriter.sql: Choisissez ce script pour installer le plug-in Rewriter et ses composants associés.

- uninstall_rewriter.sql: choisissez ce script pour désinstaller le plug-in Rewriter et ses composants associés.

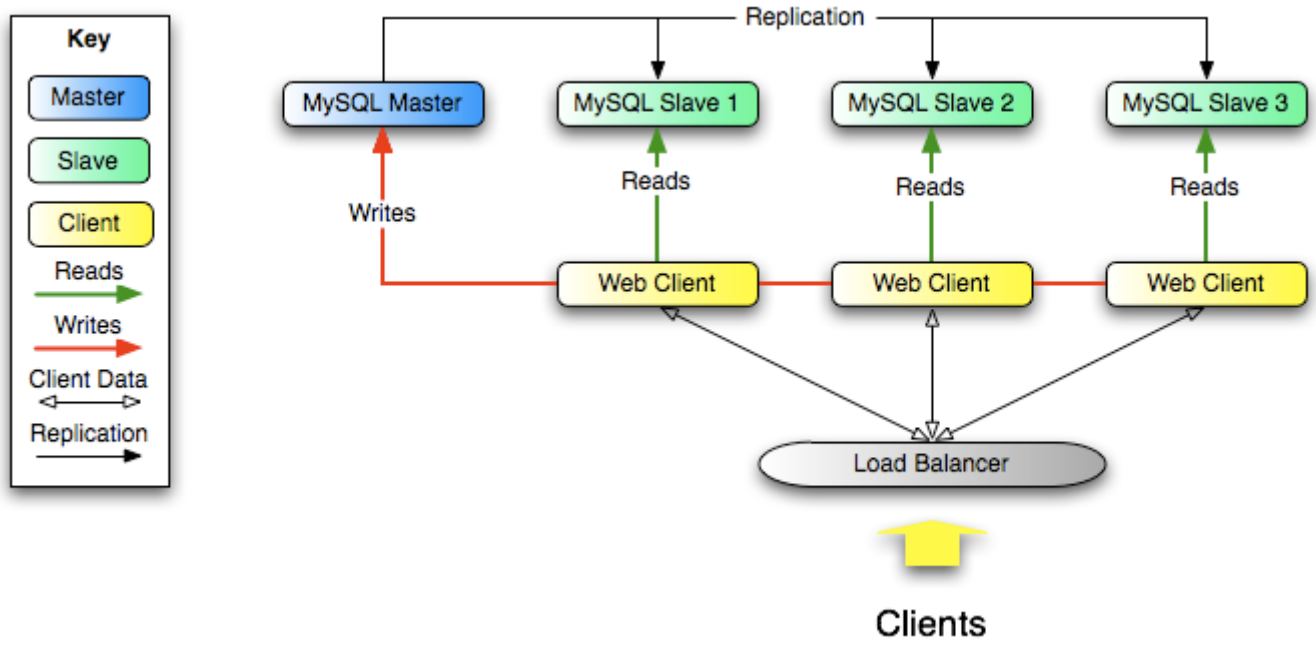
MySQL Administration

- Rappels
- Fonctions avancées de l'administration
- Réplication
  - Journal binaire et cohérence transactionnelle.
  - Formats de journalisation binaire : par instruction, par ligne, mixte.
  - Réplication simple maître-esclave. Stratégies évoluées de réplication. Détails de l'implémentation.
  - Etats des threads et fichiers de relais.
  - Options de démarrage de la réplication.
  - Résolution des problèmes courants.
- MySQL cluster

Réplication

- Mise en œuvre de la réplication

Réplication



Réplication

- La réplication permet de copier les données d'un serveur de base de données MySQL (le maître) vers un ou plusieurs serveurs de base de données MySQL (les esclaves).
- La réplication est asynchrone par défaut. Les esclaves n'ont pas besoin d'être connectés en permanence pour recevoir les mises à jour du maître.
- Selon la configuration, vous pouvez répliquer toutes les bases de données, des bases de données sélectionnées ou même des tables sélectionnées dans une base de données.

Réplication

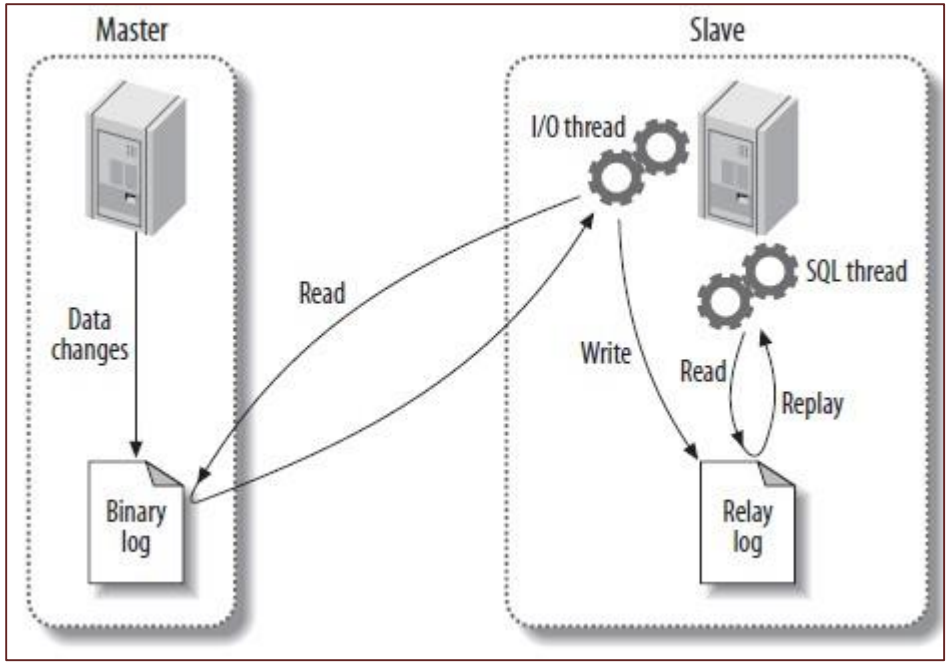
Les avantages de la réplication dans MySQL incluent:

- Solutions évolutives: répartissez la charge entre plusieurs esclaves pour améliorer les performances. Dans cet environnement, toutes les écritures et mises à jour doivent avoir lieu sur le serveur maître. Les lectures peuvent toutefois avoir lieu sur un ou plusieurs esclaves. Ce modèle peut améliorer les performances d'écriture (le maître étant dédié aux mises à jour), tout en augmentant considérablement la vitesse de lecture sur un nombre croissant d'esclaves.
- Sécurité des données: comme les données sont répliquées sur l'esclave et que l'esclave peut suspendre le processus de réplication, il est possible d'exécuter des services de sauvegarde sur l'esclave sans altérer les données principales correspondantes.
- Analytics - des données en direct peuvent être créées sur le maître, tandis que l'analyse des informations peut s'effectuer sur l'esclave sans affecter les performances du maître.
- Distribution de données longue distance - vous pouvez utiliser la réplication pour créer une copie locale des données pour un site distant, sans accès permanent au maître.

Réplication

- La réplication dans MySQL prend en charge différents types de synchronisation. Le type de synchronisation d'origine est une réplication unidirectionnelle et asynchrone, dans laquelle un serveur agit en tant que maître, tandis qu'un ou plusieurs autres serveurs agissent en tant qu'esclaves.
- Dans MySQL 8.0, la réplication semi-synchrone est prise en charge en plus de la réplication asynchrone intégrée. Avec la réplication semi-synchrone, une validation est effectuée sur les blocs maîtres avant de revenir à la session ayant exécuté la transaction jusqu'à ce qu'au moins un esclave reconnaisse avoir reçu et consigné les événements de la transaction

Réplication



Réplication

- La réplication repose sur le serveur maître qui conserve toutes les modifications apportées à ses bases de données (mises à jour, suppressions, etc.) dans son journal binaire. Le journal binaire sert d'enregistrement écrit de tous les événements modifiant la structure de la base de données ou son contenu (données) à partir du démarrage du serveur. En règle générale, les instructions SELECT ne sont pas enregistrées car elles ne modifient ni la structure de la base de données ni le contenu.
- Chaque esclave qui se connecte au maître demande une copie du journal binaire. C'est-à-dire qu'il extrait les données du maître plutôt que de les pousser vers l'esclave. L'esclave exécute également les événements à partir du journal binaire qu'il reçoit. Cela a pour effet de répéter les modifications d'origine telles qu'elles ont été effectuées sur le maître. Les tables sont créées ou leur structure modifiée, et les données sont insérées, supprimées et mises à jour en fonction des modifications apportées à l'origine sur le maître. Étant donné que chaque esclave est indépendant, la lecture des modifications du journal binaire du maître a lieu indépendamment sur chaque esclave connecté au maître.
- De plus, étant donné que chaque esclave reçoit une copie du journal binaire uniquement en le demandant au maître, il est capable de lire et de mettre à jour la copie de la base de données à son rythme et peut démarrer et arrêter le processus de réplication à sa guise sans affecter les données. la possibilité de mettre à jour le dernier état de la base de données du côté maître ou du côté esclave.

Réplication

La réplication fonctionne car les événements écrits dans le journal binaire sont lus à partir du maître, puis traités sur l'esclave. Les événements sont enregistrés dans le journal binaire dans différents formats en fonction du type d'événement. Les différents formats de réplication utilisés correspondent au format de journalisation binaire utilisé lors de l'enregistrement des événements dans le journal binaire du maître.

Format de Réplication

- Comment marche la réplication et en particulier quel est le format de transfert des données entre un maitre et un esclave

Format de Réplication

- MySQL peut utiliser l'un des trois formats pour les journaux binaires:
- STATEMENT : c'est l'instruction qui est stocké dans le journal Ce mode peut impliquer des incohérences.
- ROW : c'est la données qui est stocké dans le journal.
- MIXED : travaille comme le mode STATEMENT sauf si il est possible qu'une incohérence. Dans ce cas, MySQL bascule automatiquement en mode ROW.

Format de Réplication

La corrélation entre les formats de journalisation binaire et les termes utilisés lors de la réplication est la suivante:

Lors de l'utilisation de la journalisation binaire basée sur des instructions, le maître écrit des instructions SQL dans le journal binaire. La réplication du maître sur l'esclave fonctionne en exécutant les instructions SQL sur l'esclave. C'est ce qu'on appelle la réplication basée sur les instructions (ce qui peut être abrégé en SBR), qui correspond au format de journalisation binaire basé sur les instructions MySQL.

Lors de l'utilisation de la journalisation basée sur les lignes, le maître écrit dans le journal binaire des événements indiquant comment les rangées individuelles d'une table sont modifiées. La réplication du maître sur l'esclave fonctionne en copiant les événements représentant les modifications apportées aux lignes du tableau sur l'esclave. C'est ce qu'on appelle la réplication basée sur les lignes (qui peut être abrégée en RBR). La journalisation basée sur les lignes est la méthode par défaut. Vous pouvez également configurer MySQL pour utiliser une combinaison de journalisation basée sur des instructions et sur des lignes, en fonction de l'option la plus appropriée pour la journalisation des modifications.

Réplication

- Chaque format de journalisation binaire présente des avantages et des inconvénients. Pour la plupart des utilisateurs, le format de réplication mixte devrait fournir la meilleure combinaison d'intégrité et de performance des données. Si, toutefois, vous souhaitez tirer parti des fonctionnalités propres au format de réplication basé sur les instructions ou basé sur les lignes pour effectuer certaines tâches, vous pouvez utiliser les informations de cette section, qui résume leurs avantages et inconvénients relatifs.

Avantages de la réplication basée sur des instructions

Technologie éprouvée.

- Moins de données écrites dans les fichiers journaux. Lorsque des mises à jour ou des suppressions affectent de nombreuses lignes, l'espace de stockage requis pour les fichiers journaux est considérablement réduit. Cela signifie également que les sauvegardes et les restaurations peuvent être effectuées plus rapidement.
- Les fichiers journaux contiennent toutes les instructions ayant apporté des modifications. Ils peuvent donc être utilisés pour auditer la base de données.

Désavantages de la réplication basée sur des instructions

La totalité les instructions qui modifient des données (telles que les instructions INSERT DELETE, UPDATE et REPLACE) ne peuvent pas être répliquées à l'aide de la réplication basée sur des instructions:

Tout comportement non déterministe est difficile à reproduire lors de l'utilisation d'une réplication basée sur des instructions.

Pour InnoDB: Une instruction INSERT qui utilise AUTO_INCREMENT bloque d'autres instructions INSERT non conflictuelles.

Avantages de la réplication basée sur des lignes

- Toutes les modifications peuvent être répliquées. C'est la forme de réplication la plus sûre.
- Moins de verrous de lignes sont nécessaires sur le maître, ce qui permet d'obtenir une concurrence accrue,

Désavantages de la réplication basée sur des lignes

- RBR peut générer plus de données à consigner. Pour répliquer une instruction DML (telle qu'une instruction UPDATE ou DELETE), la réplication basée sur une instruction écrit uniquement l'instruction dans le journal binaire. En revanche, la réplication basée sur les lignes écrit chaque ligne modifiée dans le journal binaire.
- Pas de support correct des tables temporaires
- Pour les tables utilisant le moteur de stockage MyISAM, un verrouillage plus fort est requis sur l'esclave pour les instructions INSERT lors de leur application en tant qu'événements basés sur des lignes au journal binaire que lors de leur application en tant qu'instructions. Cela signifie que les insertions simultanées sur les tables MyISAM ne sont pas prises en charge lors de l'utilisation d'une réplication basée sur des lignes.

Désavantages de la réplication basée sur des lignes

- les instructions utilisant exclusivement des tables temporaires ne sont pas consignées sur le maître et, par conséquent, les tables temporaires ne sont pas répliquées.
- Un autre désavantage concerne les type BLOB/TEXT non modifié. Par défaut le mode ROW stocke dans le journal binaire l'intégralité de contenu des colonnes modifié. Si vous modifier une ligne ayant un blob non modifié, alors la totalité du blob seras inscrit dans le journal binaire.
- Il est possible d'utiliser soit un schéma différent de base de donnée soit la variable binlog_row_image

binlog_row_image

- Avec binlog_row_image =full (defaut) toutes les colonnes des lignes modifiées sont inscrites dans les journaux binaires.
- Avec binlog_row_image =minimal seules les colonnes modifiées sont inscrites
- Avec binlog_row_image =noblob toutes les colonnes sauf les champs BLOB et TEXT sont inscrites. Les champs BLOB et TEXT ne sont inscrit dans les journaux binaires que s'ils sont modifiées.

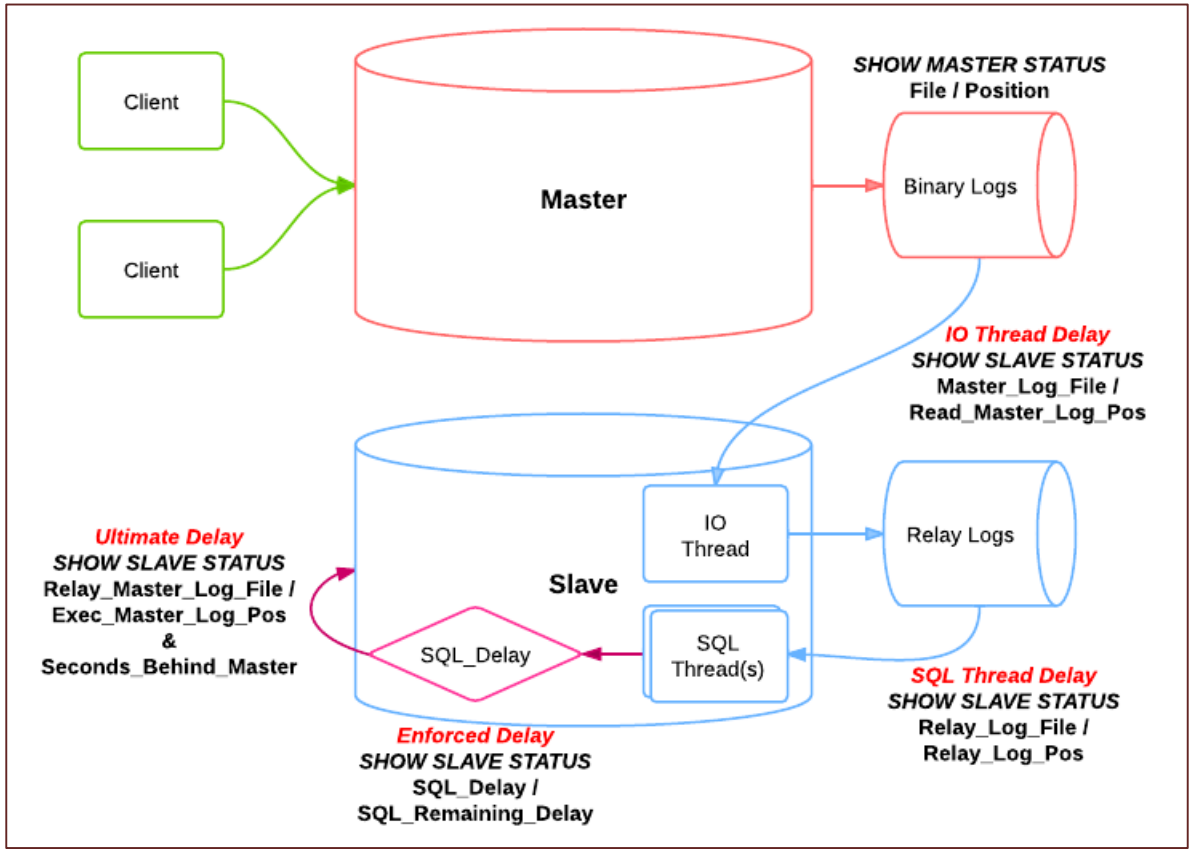
Format de Réplication

- De façon général, il est préférable d'utiliser le format ROW (plus efficace).
- Le format MIXED est a évité car buggé.

Thread de replication

- Quelles sont les différents thread de la replication?
- Comment voir l'activité des esclaves?

Thread de réplication



Thread de Réplication

- il existe trois threads par connexion maître / esclave. Un maître ayant plusieurs esclaves crée un thread de vidage de journal binaire pour chaque esclave actuellement connecté, et chaque esclave a ses propres threads d'E / S et SQL.
- Un esclave utilise deux threads pour séparer les mises à jour de lecture du maître et les exécuter en tâches indépendantes. Ainsi, la tâche de lecture des instructions n’est pas ralentie si son exécution est lente.

Thread de Réplication: BinLog Thread

Les capacités de réplication MySQL sont implémentées à l'aide de trois threads, un sur le serveur maître et deux sur l'esclave:

- Binlog dump thread. Le maître crée un thread pour envoyer le contenu du journal binaire à un esclave lors de la connexion de l'esclave. Ce thread peut être identifié dans la sortie de SHOW PROCESSLIST sur le maître en tant que thread de vidage Binlog. Le thread Binlog acquiert un verrou sur le journal binaire du maître pour lire chaque événement à envoyer à l'esclave. Dès que l'événement a été lu, le verrou est libéré, même avant que l'événement ne soit envoyé à l'esclave.

Thread de Réplication: Thread Slave

Thread Slave I / O.

Lorsqu'une instruction START SLAVE est émise sur un serveur esclave, l'esclave crée un thread d'E / S qui se connecte au maître et lui demande d'envoyer les mises à jour enregistrées dans ses journaux binaires. Le Thread Slave I / O lit les mises à jour envoyées par le thread de Binlog du maître et les copie dans des fichiers locaux qui constituent le journal de relais de l'esclave.

L'état de ce fil est indiqué comme Thread Slave I / O dans la sortie de SHOW SLAVE STATUS.

Thread de Réplication: Thread Slave SQL

- Thread Slave SQL. L'esclave crée un thread SQL pour lire le journal de relais écrit par le Thread Slave I / O esclave et exécuter les événements qu'il contient.

Réplication et Logs

Pendant la réplication, un serveur esclave crée plusieurs journaux contenant les événements de journal binaires relayés du maître à l'esclave et enregistre des informations sur l'état et l'emplacement actuels dans le journal de relais. Il existe trois types de journaux utilisés dans le processus, répertoriés ici:

- Le journal de relais comprend les événements lus dans le journal binaire du maître et écrits par le thread d'E / S esclave. Les événements du journal de relais sont exécutés sur l'esclave dans le cadre du thread SQL.
- Le journal d'informations du maître contient l'état et la configuration actuelle de la connexion de l'esclave au maître. Ce journal contient des informations sur le nom d'hôte maître, les informations d'identification de connexion et les coordonnées indiquant dans quelle mesure l'esclave a lu le journal binaire du maître.
- Le journal d'informations principal est écrit dans la table mysql.slave_master_info. Le journal d'informations du journal de relais contient des informations sur l'état du point d'exécution dans le journal de relais de l'esclave. Le journal de relais est écrit dans la table mysql.slave_relay_log_info.

Vérifier le statut de la Réplication

- Lors de la gestion d'un processus de réplication, la tâche la plus courante consiste à s'assurer que la réplication est en cours et qu'aucune erreur ne s'est produite entre l'esclave et le maître.
- L'instruction SHOW SLAVE STATUS, que vous devez exécuter sur chaque esclave, fournit des informations sur la configuration et l'état de la connexion entre le serveur esclave et le serveur maître.

Show Slave Status

```
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: master1
                  Master_User: root
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000004
          Read_Master_Log_Pos: 931
               Relay_Log_File: slave1-relay-bin.000056
                Relay_Log_Pos: 950
        Relay_Master_Log_File: mysql-bin.000004
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
```

Show Slave Status

```
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 931
              Relay_Log_Space: 1365
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids: 0
```

Principaux paramètres

- Slave_IO_State: Statut actuel de l'esclave.
- Slave_IO_Running: Indique si le thread d'E / S chargé de lire le journal binaire du maître est en cours d'exécution. Normalement, vous voulez que ce soit Oui, sauf si vous n'avez pas encore commencé la réplication ou si vous l'avez explicitement arrêtée avec STOP SLAVE.
- Slave_SQL_Running: Indique si le thread SQL d'exécution des événements dans le journal de relais est en cours d'exécution. Comme avec le thread d'E / S, cela devrait normalement être Oui. Last_IO_Error, Last_SQL_Error: dernières erreurs enregistrées par les unités d'exécution E / S et SQL lors du traitement du journal de relais. Idéalement, ils doivent être vides, sans erreur.
- Seconds_Behind_Master: nombre de secondes pendant lesquelles le thread SQL esclave est en retard pour le traitement du journal binaire principal. Un nombre élevé (ou croissant) peut indiquer que l'esclave est incapable de gérer les événements du maître en temps voulu.

Etat d'un Slave

- Checking master version :Un état qui se produit très brièvement après l'établissement de la connexion au maître.
  Connecting to master:Le thread tente de se connecter au maître.
- Queueing master event to the relay log
- Reconnecting after a failed binlog dump request
- Reconnecting after a failed master event read
- Registering slave on master
- Requesting binlog dump
- Waiting for its turn to commit
- Waiting for master to send event
- Waiting for master update
- Waiting for slave mutex on exit
- Waiting for the slave SQL thread to free enough relay log space
- Waiting to reconnect after a failed binlog dump request
- Waiting to reconnect after a failed master event read

Vérifier le statut de la Réplication

- Sur le maître, vous pouvez vérifier l'état des esclaves connectés à l'aide de SHOW PROCESSLIST pour examiner la liste des processus en cours d'exécution. Les connexions esclaves ont un vidage Binlog dans le champ Commande:

```
mysql> SHOW PROCESSLIST \G;
*************************** 4. row ***************************
     Id: 10
   User: root
   Host: slave1:58371
     db: NULL
Command: Binlog Dump
   Time: 777
  State: Has sent all binlog to slave; waiting for binlog to be
updated
   Info: NULL
```

Stopper la Réplication

- Vous pouvez arrêter et démarrer la réplication sur l'esclave à l'aide des instructions STOP SLAVE et START SLAVE. Pour arrêter le traitement du journal binaire à partir du maître, utilisez STOP SLAVE:
  - mysql> STOP SLAVE;
- Lorsque la réplication est arrêtée, le thread d'E / S esclave cesse de lire les événements du journal binaire principal et de les écrire dans le journal de relais, et le thread SQL arrête de lire et d'exécuter les événements du journal de relais. Vous pouvez mettre en pause le thread d'E / S ou SQL individuellement en spécifiant le type de thread:
  - mysql> STOP SLAVE IO_THREAD;
  - mysql> STOP SLAVE SQL_THREAD;

Redemarrer la replication

- Pour relancer l'exécution, utilisez l'instruction START SLAVE:
  - mysql> START SLAVE;
- Pour redemarrer un thread particulier
  - mysql> START SLAVE IO_THREAD;
  - mysql> START SLAVE SQL_THREAD;

Pourquoi arrêter/redemarrer la replication

- Pour un esclave qui effectue des mises à jour uniquement en traitant des événements du maître, il peut être utile d'arrêter uniquement le thread SQL si vous souhaitez effectuer une sauvegarde ou une autre tâche. Le thread d'E / S continuera à lire les événements du maître mais ils ne sont pas exécutés. Cela facilite le rattrapage de l'esclave lorsque vous redémarrez le thread SQL. En arrêtant uniquement le thread d'E / S, les événements du journal de relais peuvent être exécutés par le thread SQL jusqu'au point où le journal de relais se termine.
- Cela peut être utile lorsque vous souhaitez suspendre l'exécution pour rattraper les événements déjà reçus du maître, lorsque vous souhaitez effectuer une administration sur l'esclave, mais également vous assurer qu'il a traité toutes les mises à jour à un point spécifique. Cette méthode peut également être utilisée pour suspendre la réception d'un événement sur l'esclave pendant que vous effectuez une administration sur le maître. Arrêter le thread d'E / S mais autoriser l'exécution du thread SQL permet de s'assurer qu'il n'y a pas de retard accumulé d'événements à exécuter lorsque la réplication est redémarrée

Mise en place de la réplication

- Quelques notes de mise en œuvre de la réplication

Configuration

- Nous utiliserons ici deux serveurs distincts, l'un maitre, l'autre esclave :
- Le serveur MySQL1 (maître) : 192.168.0.21 / 255.255.255.0
- Le serveur MySQL2 (esclave) : 192.168.0.22 / 255.255.255.0

Replication Master Slave

- En premier lieu, nous allons configurer le serveur maitre en modifiant le fichier my.ini:

```
server-id = 1
log_bin = /var/log/mysql/mysql-bin.log
expire_logs_days = 10
max_binlog_size = 100M
binlog_do_db = exempledb
```

Réplication Master Slave

- Créez l'utilisateur « repuser » qui sera dédié au serveur de réplication des données

```
mysql -u root -p
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repuser'@'192.168.0.22' IDENTIFIED BY
'motdepasse';
mysql> FLUSH PRIVILEGES;
mysql> USE exempledb;
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SHOW MASTER STATUS;
mysql> exit
```

SHOW MASTER STATUS

- Les valeurs des colonnes « **File** » et « **Position** » sont à garder puisqu'elles nous serons nécessaire plus tard.
- La colonne File indique le nom du fichier journal et la colonne Position indique la position dans le fichier.
- Dans votre exemple, le fichier journal binaire est mysql-bin.000003 et la position est 73. Enregistrez ces valeurs. Vous en aurez besoin plus tard lorsque vous installerez l'esclave. Ils représentent les coordonnées de réplication auxquelles l'esclave devrait commencer à traiter les nouvelles mises à jour du maître.

Dump du master

- Faites un dump de la base données « **exempledb** » :
  - mysqldump -u root -p exempledb > exempledb.sql
- Puis désactivez le mode « **lecture seule** »
  - mysql> UNLOCK TABLES;

Serveur Esclave

- Connectez-vous sur votre serveur esclave afin d'y créer la base de données « **exempledb** » qui sera répliquée
- Importez les données du maitre grâce au dump :

```
mysql> CREATE DATABASE exempledb;
mysql> exit
mysql -u root -p exempledb < exempledb.sql
```

Configuration du serveur Esclave

- Puis configurez votre serveur esclave grâce au fichier « **/etc/mysql/my.cnf** » :

```
server-id = 2
log_bin = /var/log/mysql/mysql-bin.log
expire_logs_days = 10
max_binlog_size = 100M
binlog_do_db = exempledb
```

Configuration du serveur Esclave

- Connectez-vous au serveur MySQL afin de configurer la réplication :

```
mysql -u root – p
mysql> CHANGE MASTER TO MASTER_HOST='192.168.0.21',
MASTER_USER='repuser', MASTER_PASSWORD='motdepasse',
MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS= 340;
mysql> START SLAVE;
```

Force et Faiblesse de la configuration

La configuration maître esclave permet d'être évolutif en lecture. Plus le nombre d'esclave est élevé, plus le nombre de lecture est importante.

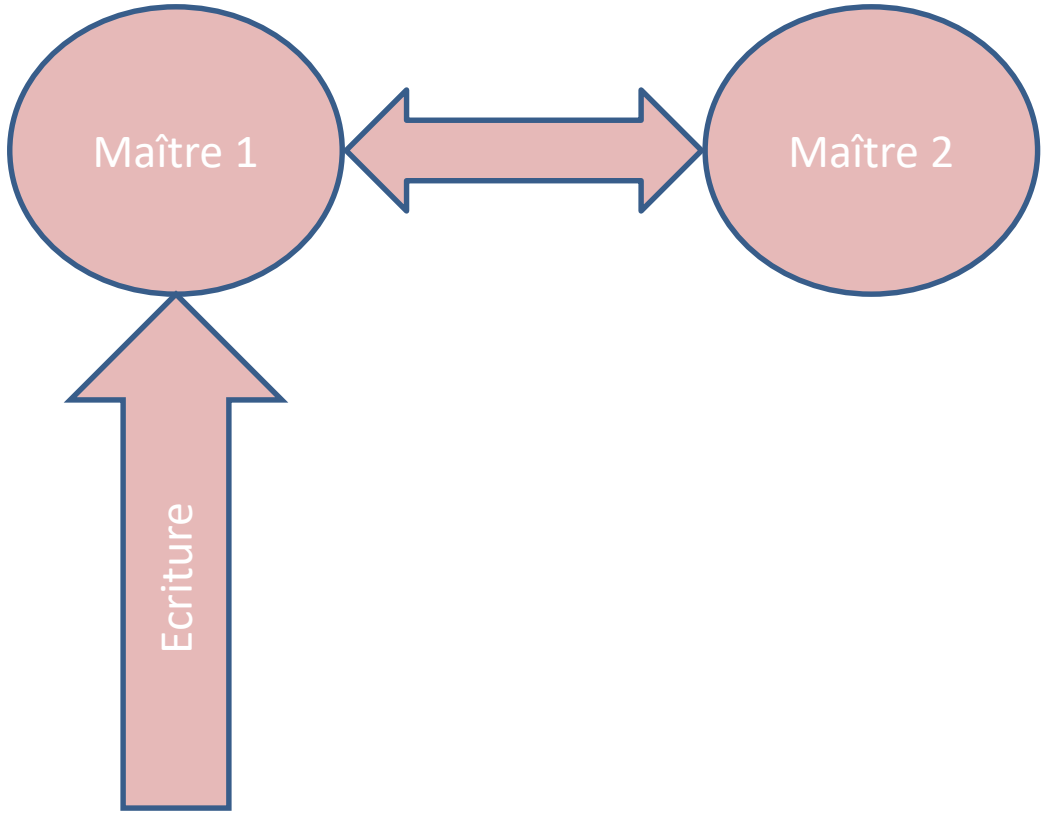
Par default, MySQL n'empêche par les écritures sur les esclaves, mais il est possible et conseillé de positionner la variable read-only pour empêcher les modifications.

La réplication asynchrone ne certifient pas que les données des escalves sont synchrone avec les données du maître. Si la fraîcheur des données est impératif, la requête de selection devras se faire sur le maître

Réplication maître-maître

- Comment faire pour avoir plusieurs maître?
- Qu'elle est le but d'une réplication master-master?

Master-Master Replication



Configuration

La configuration d'une replication bidirectionnelle se fait seimplement en configurant chaque serveur à la fois comme un maître et comme un escalve. Il est recommandé d'ajouter l'option log_slave_updates=1 dans le fichier de configuration de chaque serveurs.

Le but de cette option est de s'assurer que le serveur qui a le rôle d'esclave écrait dans ses journaux binaires les requêtes provenant du serveur qui a le rôle de maître.

Avec cette option vous pourrez alors ajouter des esclaves à n'importe lequel des deux serveurs

Configuration

Si le serveurs A et B sont configurés avec une réplication bidirectionnelle, une écriture executée sur A seras répliquée vers B, mais elle ne sera pas répliquée une seconde fois de B vers A.

Chaque requête stocke dans les journaux binaires son serveur d'origine et la réplication ne va jamais exécuter sur A par la réplication d'une écriture provenant de B mais marquée comme ayant été exécutée par A.

Avantages/Inconvénients

Le cas le plus simple est le mode actif-passif dans lequel un maître est utilisé pour les lectures/écritures tandis que le second maître est utilisé en tant que maître de secours.

Cela permet d'éviter tout conflit au niveau de la réplication puisque les modifications ne sont faites que sur un serveur.

Il est aussi possible d'utiliser une configuration multi-maitre en mode actif-actif en permettant aux lectures/écritures d'être faites sur chacun des maîtres.

Néanmoins dans cette configuration, il est essentiel de gérer les conflits.

Configuration actif-actif

Il y a trois types de conflit en mode actif-actif:

- Les champs auto-incréments doivent être gérer. En effet, si une insertion a lieu en même temps sur deux serveurs, le même ID seras mis pour 2 requete différentes.
    - L'astuce consiste a modifier les variables auto_increment_increment qui indique le pas à utiliser pour les auto increment et le champ auto increment offset.
- Les champs uniques doivent être gérer en particulier si ils sont dépendants d'une opération précédente dans la base (insertion dans une liste chaîné par exemple).
- Enfin les opérations non commutative doivent être gérer:
    - Update table set a=a*2, update table set a=a-10

Scénario de réplication

- Comment utiliser la réplication dans un but de backup ou de performance

Réplication pour les backups

Pour utiliser la réplication comme solution de sauvegarde, répliquez les données du maître sur un esclave, puis sauvegardez l'esclave de données. L'esclave peut être suspendu et arrêté sans affecter le fonctionnement du maître. Vous pouvez ainsi produire un instantané efficace des données «en direct» qui nécessiteraient sinon la fermeture du maître. La façon dont vous sauvegardez une base de données dépend de sa taille et du fait que vous sauvegardiez uniquement les données, ou les données et l'état de l'esclave de réplication, afin de pouvoir reconstruire l'esclave en cas de défaillance.

Réplication pour les backups

- Si vous utilisez la réplication comme solution pour vous permettre de sauvegarder les données sur le maître, et que la taille de votre base de données n'est pas trop grande, l'outil mysqldump peut convenir
- Pour les bases de données plus grandes, où mysqldump serait peu pratique ou inefficace, vous pouvez sauvegarder les fichiers de données brutes à la place. L'utilisation de l'option de fichiers de données brutes signifie également que vous pouvez sauvegarder les journaux de relais et binaires qui vous permettront de recréer l'esclave en cas de défaillance de l'esclave.

Backup via MySQLDump

- Utiliser mysqldump pour créer une copie d'une base de données vous permet de capturer toutes les données de la base de données dans un format permettant d'importer les informations dans une autre instance de MySQL Server
- Arrêtez l'esclave de traiter les demandes. Vous pouvez arrêter uniquement le thread SQL esclave pour suspendre l'exécution de l'événement:
  - shell> mysql -e 'STOP SLAVE SQL_THREAD;
- Exécutez mysqldump pour vider vos bases de données
  - shell> mysqldump --all-databases > fulldb.dump'
- Redemarrer le thread SQL
  - mysql> START SLAVE SQL_THREAD;

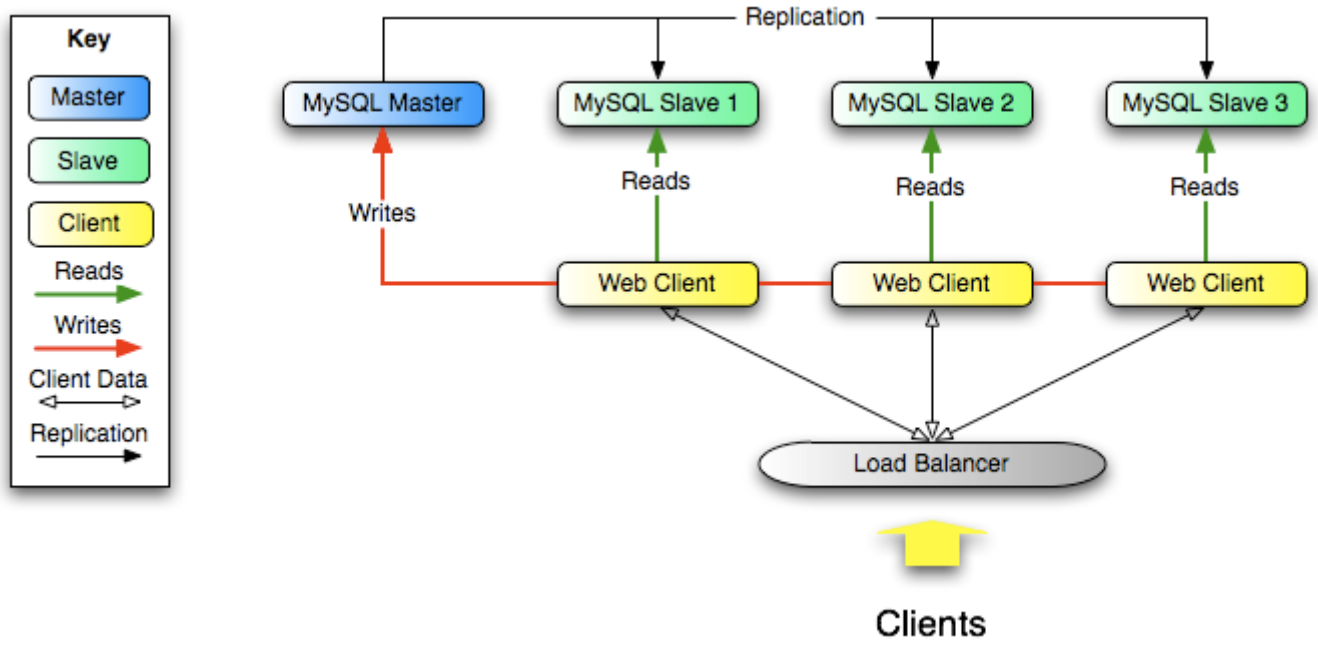
Backup par copy

- Pour garantir l'intégrité des fichiers copiés, la sauvegarde des fichiers de données brutes sur votre esclave de réplication MySQL doit avoir lieu lorsque votre serveur esclave est arrêté.
  - Arreter le server shell> mysqladmin shutdown
  - Copier les données via un tar-gz du repertoire data de mysql
  - Redemarrer le serveur MySQL

Réplication pour la monté en charge

- Vous pouvez utiliser la réplication en tant que solution de monté en charge; c'est-à-dire que vous souhaitez répartir la charge de requêtes de base de données sur plusieurs serveurs de base de données, dans des limites raisonnables.
- Etant donné que la réplication fonctionne de la distribution d'un maître à un ou plusieurs esclaves, son utilisation pour la montée en puissance parallèle fonctionne mieux dans un environnement où le nombre de lectures est élevé et le nombre d'écritures / mises à jour est faible.
- La plupart des sites Web entrent dans cette catégorie, où les utilisateurs le parcourent, lisent des articles, publient des articles ou consultent des produits…

Réplication pour la montée en charge



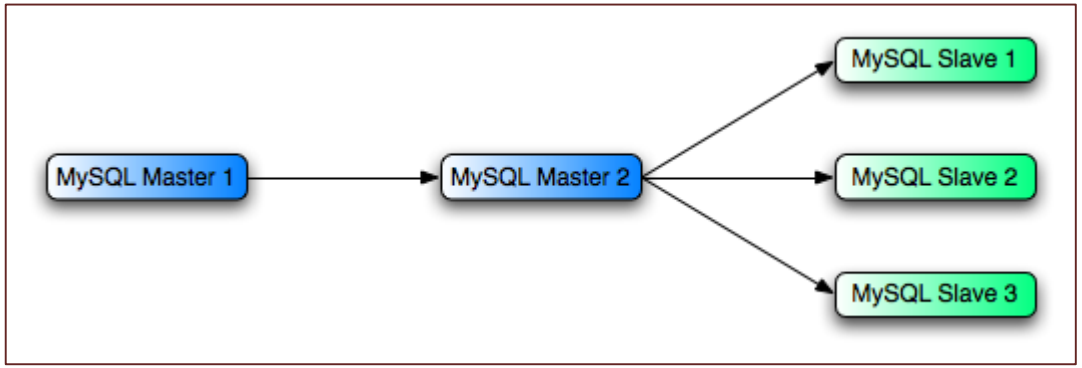
Réplication pour la montée en charge

- Modifiez l'implémentation de votre accès à la base de données pour envoyer toutes les écritures au maître et pour envoyer des lectures au maître ou à l'esclave.

Amélioration des performances

- À mesure que le nombre d'esclaves se connectant à un maître augmente, la charge augmente, chaque esclave utilisant 1 thread master et 1 connexion réseau.
- De plus, comme chaque esclave doit recevoir une copie complète du journal binaire du maître, la charge du réseau sur le maître peut également augmenter et créer un goulot d'étranglement.
- Une solution est d'avoir un serveur MySQL dédié à la réplication et 1 serveur dédié aux écritures

Architecture Multi Master



Multi Master Architecture

Pour que cela fonctionne, vous devez configurer les instances MySQL comme suit:

- Maître 1 est le maître principal où toutes les modifications et mises à jour sont écrites dans la base de données. La journalisation binaire est activée sur les deux maîtres, ce qui est la valeur par défaut.
- Le maître 2 est l'esclave du maître 1 qui fournit la fonctionnalité de réplication au reste des esclaves de la structure de réplication. Master 2 est la seule machine autorisée à se connecter à Master 1. Master 2 a l'option --log-slave-updates mise à jour (qui est l'option par défaut). Avec cette option, les instructions de réplication du maître 1 sont également écrites dans le journal binaire du maître 2 afin qu'elles puissent ensuite être répliquées sur les vrais esclaves.
- Esclave 1, Esclave 2 et Esclave 3 servent d'esclaves au maître 2 et répliquent les informations du maître 2, qui consiste en réalité en les mises à jour enregistrées sur le maître 1.

Principales variables

- Il s'agit d'un rappel des principales variables de la réplications

Principales variables

Variable	Description	Rôle	Indispensable
Server-id	Identifiant du serveur	Maître/Esclave	Oui
Log-bin	Force l'ecriture dans les journaux binaires des ordres executés sur le serveur MySQL	Maître(possibl e sur l'esclave)	Oui
Log-slave-updates	Force l'écriture dans les journaux binaires des ordres reçus d'un maître et appliqués.	Esclave	Non, excepté dans une réplication circulaire ou dans une arborescence d'esclave
expire_log_days	Spécifie la rétention en nombre de jours des journaux binaires. Si cette variable est à 0, la rétention est infinie, et il faut purger les journaux via PURGE BINARY LOG	Maître/Esclave	Recommandé

Principales variables

Variable	Description	Rôle	Indispensable
Binlog-do-db,bin-log-table,binlog-ignore-db,binlog-ignore-table	Spécifie le nom d'une base/table à suivre ou à ignorer dans les journaux binaires	Maître	Non
Replicate-do-db,replicate-ignore-db, replicate-..	Spécifie le nom d'une base/table ou le modèle de nom de table à répliquer ou à ignorer sur l'esclave	Esclave	Non
Innodb_support_xa	Active le support des transactions distribuées et force le flush des logs binaires à chaque commit	Maître	Plus que recommandé
Sync_binlog=1	Force l'écriture des journaux binaires à chaque instructions autocommitée ou à chaque transaction	Maître	Non

Type de replication

- Présentation des grands types de réplication

Replication asynchrone

- La réplication asynchone est celle mise en place par défaut dans MySQL.
- Avec la réplication asynchrone, le maître écrit les événements dans son journal binaire et les esclaves les demandent lorsqu'ils sont prêts. Il n'y a aucune garantie qu'un événement atteindra un esclave.
- La réplication asynchrone signifie que le maître confirmera les transactions aux clients sans attendre que ce changement se synchronise avec les esclaves. Si le maître échoue avant la réplication de la transaction, le logiciel client et l'utilisateur estiment qu'une transaction a été validée en toute sécurité et est maintenant totalement perdue.

Replication semi-synchrone

- Le maître ne confirme pas les transactions tant qu'au moins un esclave n'a pas copié la modification dans son journal de relais. Comme la réplication synchrone, le client est convaincu que les transactions confirmées peuvent survivre à une défaillance d'un serveur unique (la modification existe sur au moins un autre serveur au moment de la confirmation). Comme la réplication asynchrone, le client obtient rapidement cette confirmation, sans attendre que chaque esclave traite la modification.

Replication semi-synchrone

Les principaux avantages de la réplication semi-synchrone sont les suivants:

- Il améliore l'intégrité des données pour les pannes sur un seul serveur.
- C'est une simple amélioration de la réplication asynchrone, pas une toute nouvelle compétence à apprendre.
- Cela n'affecte en rien les performances de lecture.
- Le maître revient à la réplication asynchrone si tous les esclaves sont en panne.

Les principaux inconvénients de la réplication semi-synchrone sont les suivants:

- Les retards réseau et d'E / S peuvent considérablement augmenter le temps d'attente du client.
- L'esclave reconnaît uniquement que la transaction a été copiée dans son journal de relais, et non appliquée à sa base de données. Cela ne résout pas les éventuels problèmes de cohérence avec les esclaves en retard de lecture.
- Le maître attend seulement qu'un esclave acquitte. Pour promouvoir un esclave pour remplacer un maître défaillant, vous devez toujours déterminer quel esclave était le plus à jour.

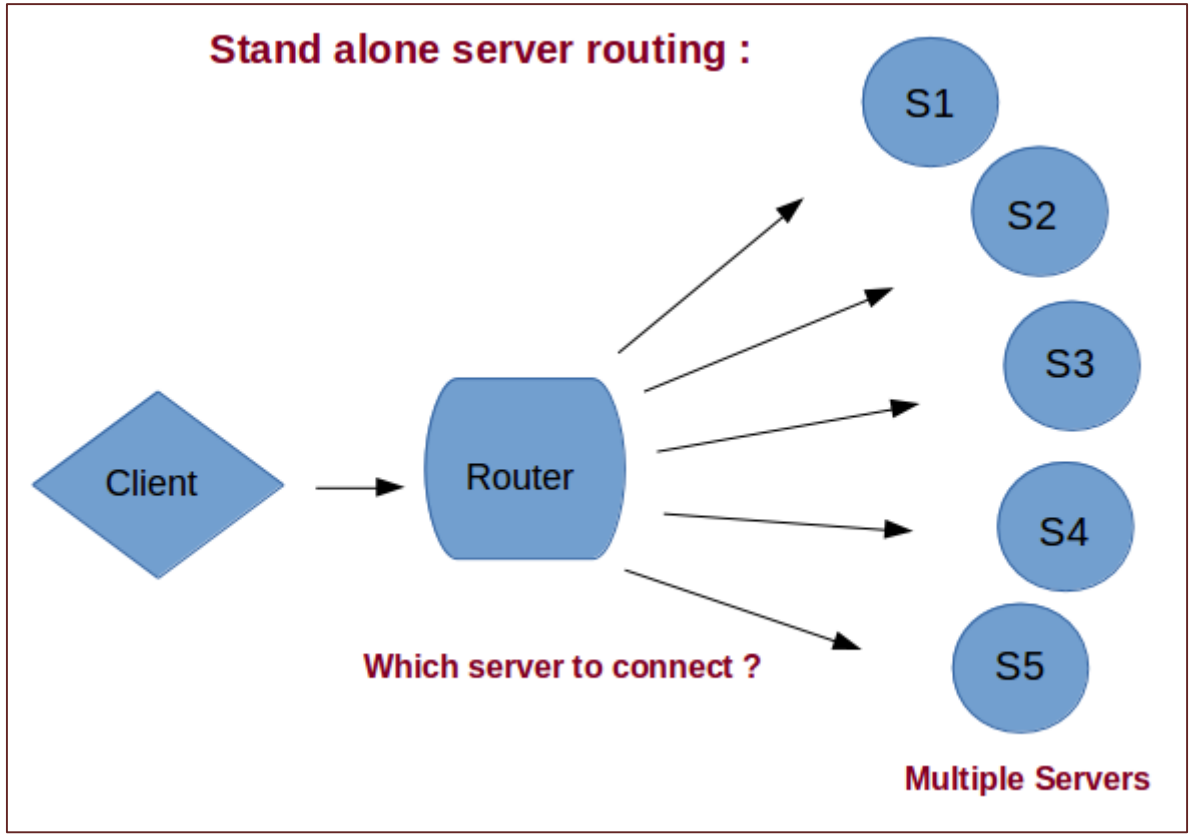
MySQL Router

- Présentation de l'outil MySQL Router pour la réplication

MySQL Router

- Pour que les applications client gèrent le basculement, elles doivent connaître la topologie du cluster InnoDB et savoir quelle instance MySQL est PRIMARY. Bien qu'il soit possible aux applications d'implémenter cette logique, MySQL Router peut fournir et gérer cette fonctionnalité.

MySQL Router



Configuration

- section logger: définissez le niveau de journalisation [FATAL, ERROR, WARNING, DEBUG et INFO]. Si non présent, le niveau INFO sera utilisé par défaut.
- section de routage: définissez l'adresse à laquelle le routeur écoutera et les serveurs à gérer, avec l'un des deux modes possibles: «lecture seule» ou «lecture-écriture». être utilisé par défaut.

Configuration

```
[logger]
level = INFO

[routing:read_only]
bind_address = localhost
bind_port = 7001
destinations = localhost:13002,localhost:13003,localhost:13004
mode = read-only

[routing:read_write]
bind_address = localhost
bind_port = 7002
destinations = localhost:13005,localhost:13006
mode = read-write
```

Configuration

[routing : read_only]:

Si vous connectez un client au service « routing : read_only », à savoir le port 7001, le routeur redirigera la première connexion vers le premier serveur disponible de la liste, à savoir 13002. Si vous connectez un autre client au même service, le routeur redirigera la connexion vers le serveur. prochain serveur disponible dans la liste, c'est-à-dire 13003.

[routing : read_write]:

Si vous connectez un client au service de routage en [routing : read_write] , à savoir le port 7002, le routeur redirigera toujours la connexion vers le premier serveur disponible de la liste, à savoir 13005. Si vous connectez un autre client au même service, le routeur redirigera la connexion vers premier serveur disponible dans la liste, soit 13005.

Problèmes courants

- Empêcher la réplication
- Non réplication de reqûetes
- Eviter les retards de réplication.
- Corriger une erreur de réplication
- Récupérer l'espace disque

Empêcher la réplication

Par défaut, toutes les écritures du maître sont écrites dans les journaux binaires.

Ces journaux sont copiés sur les esclaves, donc les esclaves répliquent toutes les requêtes du maître. Cependant dans certains cas, il n'est pas souhaitable de répliquer toutes les requêtes et en particulier si un des schémas ne doit pas être répliquer.

Le filtrage est possible sur le maître via l'options binlog-do-db et binlog-ignore-db, ou sur les esclaves via replicate-do-db et replicate-ignore-db.

Sauf exception, il est préférable de maître les filtres sur les esclaves.

Empêcher la réplication

Ces règles de filtrage ne fonctionnent pas de la même manière selon le format des journaux binaire.

En mode Statement seule la base cournate est vérifiée.

Si vous utilisez binlog-do-db=db1 l'ordre d'INSERT suivant sera répliqué:

USE db1

INSERT into db2.t1 values (100)

Inversement la requete suivante ne seras pas répliqué malgrès le faite que t2 appartiennent db1

USE db2

INSERT INTO db1.t2 values (10);

Empêcher la réplication

En mode ROW les ordres de modification sont répliqués si les objets aux-quels il s'appliquent sont concernées par les règles. Si la table modifiée n'appartienent pas à la base db1, alors elle ne sera pas répliquée.

Dans la requete suivante, seul les modifications de la table t1 seront répliquées en mode ROW

USE db2

UPDATE db1.t1,db2.t2 set i=1232,j=4310.

Empêcher la réplication

Il est possible de demander explicitement au maître de ne pas enregistrer une écriture dans les journaux binaire en utilisant l'ordre suivant dans la session qui effectue les requêtes a ne pas répliquer:

SET SQL_LOG_BIN=0

Eviter les retards de réplication

La réplication est monothreadé sur le esclaves alors que le maître peut executer des écritures en parallèle.

Une esclave souffre donc d'un handicap, et si le nombre d'écriture est imporant alors il est possible que l'esclave ait un retard important sur le maître.

Ce retard est visible via la commande SHOW SLAVE STATUS et en particulier sur la valeur Seconds_Behind_Master.

Pour réduire se retard, il est possible:

- De changer le matériel sur les esclaves en privilégiant la fréquence processeur sur le nombre de cœur. Il est aussi possible de changer le disque esclave par des SSD.
- D'utiliser le mode ROW sur le mode Statement.
- D'optimiser les requêtes

Corriger une erreur de réplication

Pour vérifier si un esclave réplique toujours les données de son maître on utilise la commande SHOW SLAVE STATUS. Si le IO_TREAD ou le SQL_THREAD ne fonctionnent pas, il faut vérifier le champ Last_Error, Last_IO_Error et Last_SQL_Error

Corriger une erreur de réplication

```
mysql> SHOW SLAVE STATUS \G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 1.2.3.4
                  Master_User: slave_user
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000024
          Read_Master_Log_Pos: 933201702
               Relay_Log_File: mysqld-relay-bin.000113
                Relay_Log_Pos: 63519994
        Relay_Master_Log_File: mysql-bin.000021
             Slave_IO_Running: Yes
            Slave_SQL_Running: No
              Replicate_Do_DB:
          Replicate_Ignore_DB: mysql,information_schema,performance_schema,test
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 1054
                   Last_Error: Error 'Unknown column 'tx_feuserbranch_agb' in 'field list'' on
query. Default database: 'dbtest'.
Query: 'INSERT INTO fe_users
(email,first_name,last_name,address,ip,city,country,telephone,fax,company,tx_feuserbranch_branc
ha,
tx_feuserbranch_customernr,tx_feuserbranch_agb,username,password,name,usergroup,disable,by_invi
tation,tx_srfeuserregister_password,
tstap,crdate,cruser_id,fe_cruser_id,pid) VALUES
('test333@example.com','John','Doe','test','55555','test','Deutschland','49111111111',
'','test','0','55','on','test333@example.com','xxxxxxxxxxxxxx','John
Doe','0','1','0','','1361359747','1361359747','0','0','33')'
```

Corriger une erreur de réplication

Dans cet exemple, la requête SQL n'a pas été répliqué à cause d'un problème de colonne, c'est pourquoi le thread SQL_THREAD c'est arrêté.

Une solution est de corriger l'erreur et de redémarrer la réplication avec la commande START SLAVE SQL_THREAD.

Parfois, après avoir corrigé l'erreur, il est souhaitable d'annuler la requête qui avait provoqué l'erreur.

Pour cela il existe la commande SET GLOBAL SQL_SLAVE_SKIP_COUNTER.

- SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1
- START SLAVE

Corriger une erreur de réplication

Vous pouvez utiliser la requête SET GLOBAL SQL_SLAVE_SKIP_COUNTER. Cette méthode indique ne pas exécuter une requête.

Récupérer l'espace disque des journaux binaires

Par défaut, les journaux binaires se sont jamais supprimés des disques.

Sans action, il finiront par être totalement remplis. Pour éviter ce problème, le plus simple est de configurer une purge automatique des journaux après N jours via l'option expire_logs_days.

Il existe aussi la commande PURGE BINARY LOG qui supprime tous les journaux binaires.

Il est possible de préciser une date ou une serie de fichier a supprimer.

MySQL Administration

- Rappels
- Fonctions avancées de l'administration
- Réplication
- MySQL cluster
    - Installation d'un cluster MySQL.
    - Architecture. Configuration hardware et système.
    - Partitionnement des tables et répartition des données sur les nœuds du cluster.
    - Le moteur NDB. Processus et fichiers. Serveur de gestion. Mise en oeuvre et administration.
    - Sauvegardes, restaurations. Cluster et réplication.
    - Limitations et évolutions.

MySQL Cluster

- Installation d'un cluster MySQL.
- Architecture. Configuration hardware et système.
- Partitionnement des tables et répartition des données sur les nœuds du cluster.
- Le moteur NDB. Processus et fichiers. Serveur de gestion. Mise en œuvre et administration.
- Sauvegardes, restaurations. Cluster et réplication.
- Limitations et évolutions.

Type de cluster SQL

- Shared nothing :
  - Chaque nœud est autonome et possède son propre disque et sa mémoire. Cela implique qu'il n'y a aucun accès disques concurrents à partir de plusieurs nœuds. Dans le cas de MySQL Cluster, une réplication synchrone des mises à jour est effectuée entre les nœuds.
- Shared disk :
  - Les nœuds possèdent chacun leur mémoire mais ils partagent une ou plusieurs ressources de stockage. Elle utilise un accès centralisé aux disques à partir de tous les nœuds. Tous les nœuds pouvant écrire de manière de concurrente via le cache disque, un mécanisme de synchronisation est nécessaire pour préserver la cohérence des données : c'est un manager de verrous distribués qui assume ce rôle.
- Shared everything :
  - Les nœuds partagent une ou plusieurs ressources de stockage. Elle utilise un cache de données dit « commun » : un mécanisme (dit de cache fusion sur Oracle RAC) a été implémenté afin que la mémoire physiquement distincte de chaque nœud soit vue comme un tout par chaque nœud.
  - Ce type de cluster permet un haut niveau de disponibilité car si un nœud est inaccessible les autres ne sont pas affectés, et de hautes performances car le cache de données commun permet de réduire les accès disques.

Installation de MySQL Cluster

- MySQL Cluster est la solution de clustering MySQL s'appuyant sur une réplication synchrone permettant de mettre en place la technique de sharding.

- MySQL Cluster est conçu pour ne pas avoir de point de défaillance unique. Dans un système sans partage, chaque composant doit posséder sa propre mémoire et son propre disque, et l'utilisation de mécanismes de stockage partagés tels que les partages réseau, les systèmes de fichiers réseau et les réseaux de stockage n'est pas recommandée ni prise en charge.
- MySQL Cluster intègre le serveur MySQL standard à un moteur de stockage en cluster en mémoire appelé NDB

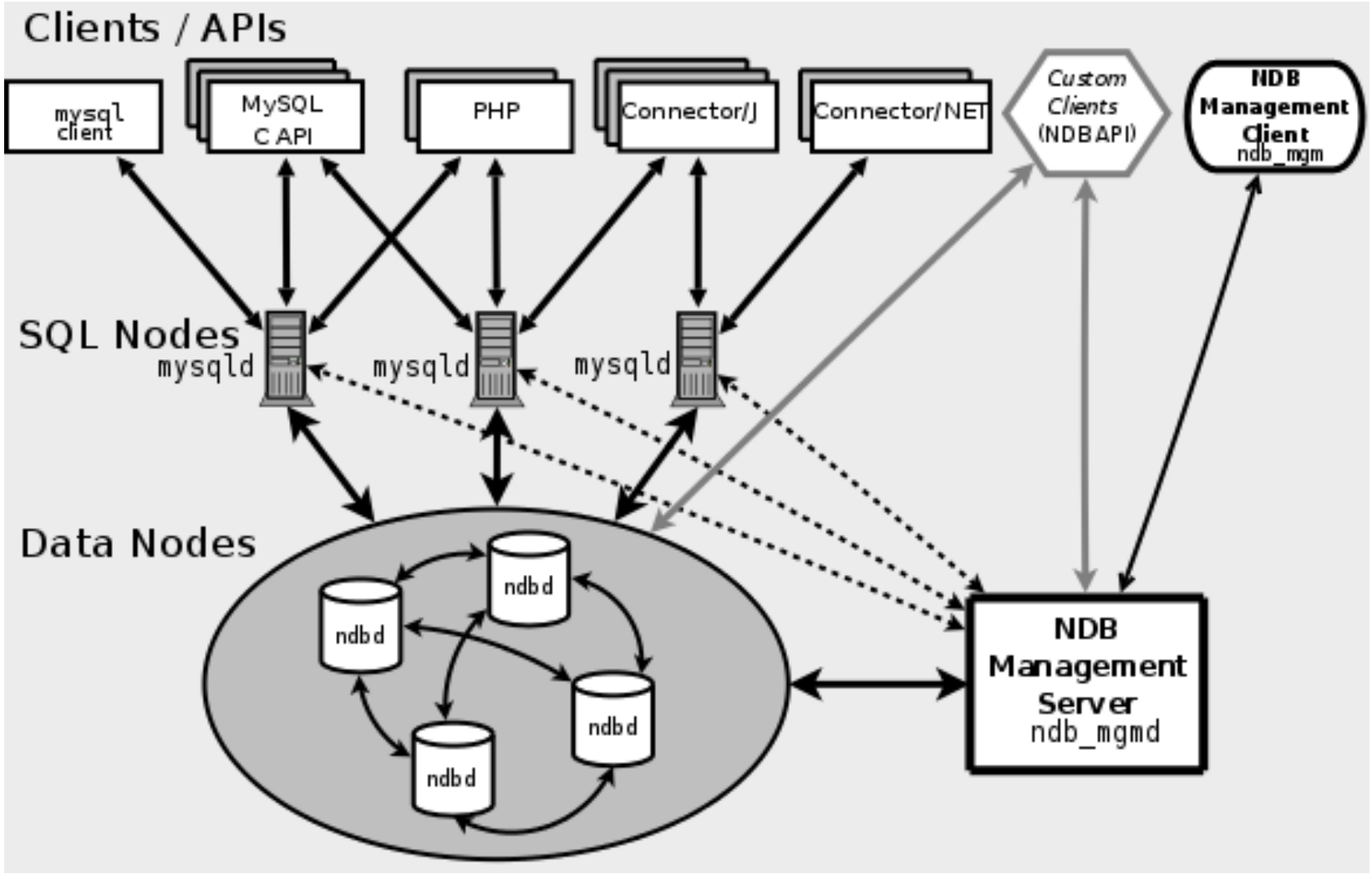
MySQL Cluster

- Un cluster MySQL est constitué d'un ensemble d'ordinateurs, appelés hôtes, exécutant chacun un ou plusieurs processus. Ces processus, appelés nœuds, peuvent inclure des serveurs MySQL (pour accéder aux données NDB), des nœuds de données (pour stocker les données), un ou plusieurs serveurs de gestion et éventuellement d'autres programmes spécialisés d'accès aux données

Type de Nœud dans le cluster

- **Les nœuds d'applications** (**sql node**) sont ceux par lesquelles passent toutes demandes d'accès aux données, il parse l'ordre SQL, détermine le coordinateur de transaction…cela peut être un serveur mysql ou une application exploitant l'api ndb.
- **Les nœuds de gestion** (**management node**) sont chargés de loguer les évènements du cluster, d'effectuer le rôle d'arbitre, arrêter/démarrer les nœuds de données, effectuer les sauvegardes…On utilise le client de gestion.
- **Les nœuds de données (data node)** sont les nœuds principaux du cluster et sont dotés des fonctionnalités suivantes : Stockage et gestion des données, Partitionnement , Réplication synchrone, …

Schema MySQL Cluster



MySQLCluster

- Bien qu'un nœud SQL MySQL Cluster utilise le démon de serveur mysqld Les données stockées dans les nœuds de données pour MySQL Cluster peuvent être mises en miroir.
- le cluster peut gérer les défaillances de nœuds de données individuels sans autre impact qu'un petit nombre de transactions est abandonné en raison de la perte de l'état de la transaction. Étant donné que les applications transactionnelles sont censées gérer les échecs de transaction, cela ne devrait pas être une source de problèmes.
- Les nœuds individuels peuvent être arrêtés et redémarrés, puis rejoindre le système (cluster). Les redémarrages successifs (dans lesquels tous les nœuds sont redémarrés) sont utilisés pour effectuer des modifications de configuration et des mises à niveau logicielles

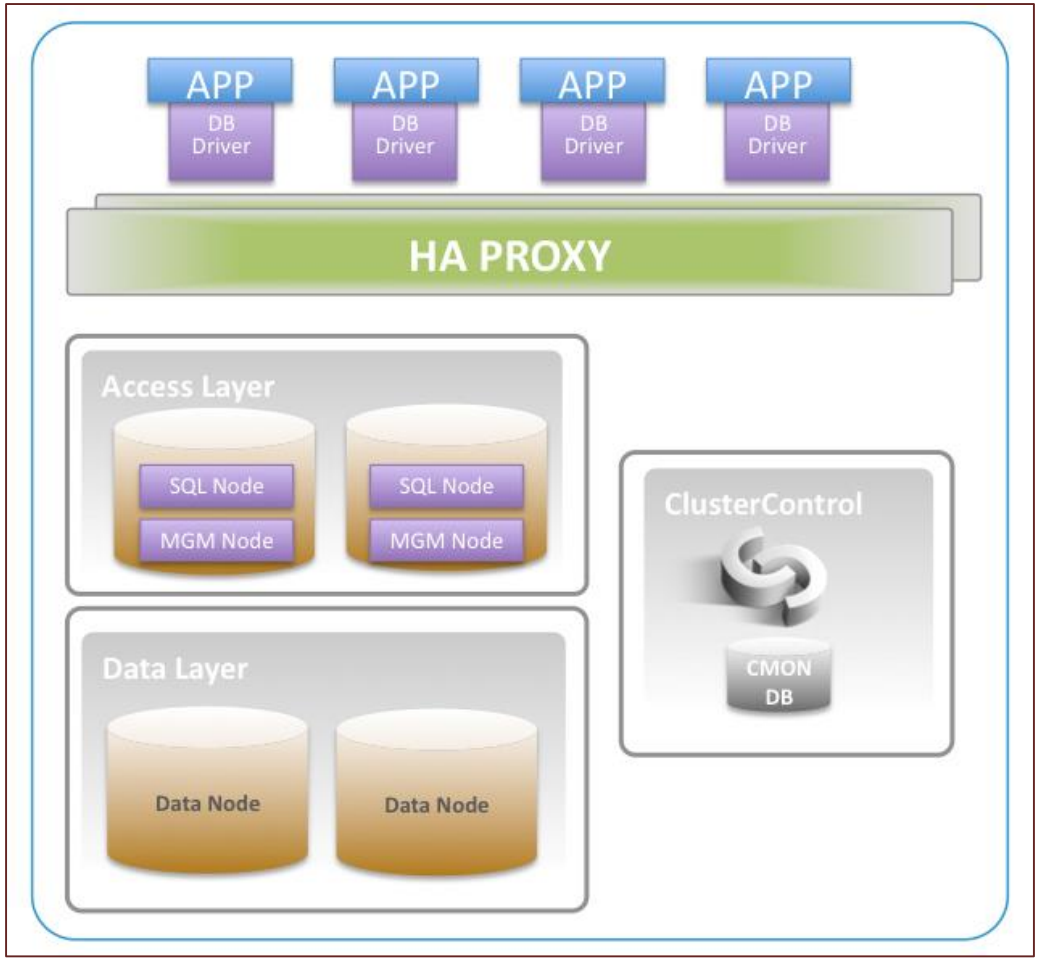
Configuration Hardware

- L'un des atouts de NDB Cluster est qu'il peut être exécuté sur du matériel standard et qu'il n'a pas d'exigences inhabituelles à cet égard, sauf pour de grandes quantités de mémoire RAM, du fait que tout le stockage de données en direct est effectué en mémoire.
- Pour la communication entre les nœuds, le cluster NDB prend en charge la mise en réseau TCP / IP dans toute topologie standard.

Configuration Hardware

- Sécurité. Les communications entre les nœuds de cluster NDB ne sont cryptées ni blindées.
- Efficacité. La configuration d'un cluster NDB sur un réseau privé ou protégé permet au cluster d'utiliser exclusivement la bande passante entre ses hôtes. L'utilisation d'un commutateur distinct pour votre cluster est vivement conseillé.
- Latence. Le cluster NDB requiert une communication entre les nœuds de données et les nœuds SQL afin d'exécuter des requêtes et des mises à jour. La latence de communication entre ces processus peut affecter directement les performances observées et la latence des requêtes des utilisateurs.(Lorsque vous définissez des valeurs de délai d'expiration telles que HeartbeatIntervalDbDb et HeartbeatIntervalDbApi pour le cluster NDB, vous devez veiller à effectuer une détection, un basculement et une remise en service rapides, tout en évitant des faux positifs)

Architecture standard basé HA Proxy



Sizing de ndb

- Vous pouvez évaluez grossièrement le taux de mémoire nécéssaire
  - (SizeofDatabase × NumberOfReplicas × 1.1) / NumberOfDataNodes
- Lors du calcul des besoins en mémoire de cluster, vous pouvez trouver l'utilitaire ndb_size.pl disponible dans les dernières versions de MySQL 5.7. Ce script Perl se connecte à une base de données MySQL actuelle (non Cluster) et crée un rapport sur l'espace requis par cette base si elle utilisait le moteur de stockage NDBCLUSTER.

Mise en oeuvre

- Comment mettre en place un cluster

Installation de MySQLCluster

- Pour l'installation de MySQL Cluster, les version des logiciels doivent être installées sur les serveurs:
- Python 2,6+
- Paramiko 1,7,7,1+
- Pycrypot version 2.6
- Libaoi
- Libnuma
- Perl-modules

Installation des packages

```
apt-get install libaio-dev
apt-get install libnuma-dev
apt-get install perl-modules
```

Configuration

Dans cette formation, nos nœuds de cluster ont les adresses IP privées suivantes:

- 198.51.100.0 sera le premier nœud de données MySQL Cluster
- 198.51.100.1 sera le deuxième nœud de données
- 198.51.100.2 sera le nœud du gestionnaire de cluster et du serveur MySQL

Installation du manager du cluster

- Pour installer le gestionnaire de cluster, nous devons d'abord récupérer le fichier d'installation .deb approprié sur la page de téléchargement officielle de MySQL Cluster.

```
wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster-community-management-server_7.6.6-1ubuntu18.04_amd64.deb
sudo dpkg -i mysql-cluster-community-management-server_7.6.6-1ubuntu18.04_amd64.deb
```

Configuration du manager du cluster

- Le gestionnaire de cluster doit être le premier composant lancé dans un cluster MySQL. Il nécessite un fichier de configuration, transmis en tant qu'argument à son exécutable. Nous allons créer et utiliser le fichier de configuration suivant: /var/lib/mysql-cluster/config.ini.

```
sudo mkdir /var/lib/mysql-cluster
sudo nano /var/lib/mysql-cluster/config.ini
```

Config.ini minimaliste

```
[ndbd default]
# Option pour les process ndbd a disposition des nœuds de données
NoOfReplicas=2  # Nombre de noeud

[ndb_mgmd]
# Option pour le process de management
hostname=198.51.100.2 # hostname du noeud
datadir=/var/lib/mysql-cluster  # Repertoire de data

[ndbd]
hostname=198.51.100.0 # IP du noeud
NodeId=2            #  identifiant unique pour le nœud de données
datadir=/usr/local/mysql/data # Repertoire de data

[ndbd]
hostname=198.51.100.1 # IP du noeud
NodeId=3            # identifiant unique pour le nœud de données
datadir=/usr/local/mysql/ # Repertoire de data

[mysqld]
# Nœud SQL:
hostname=198.51.100.2
```

Démarrage du serveur de management

- Nous pouvons maintenant démarrer le gestionnaire en exécutant le binaire ndb_mgmd et en spécifiant son fichier de configuration à l'aide de l'indicateur -f:

```
sudo ndb_mgmd -f /var/lib/mysql-cluster/config.ini
MySQL Cluster Management Server mysql-5.7.22 ndb-7.6.6
2018-07-25 21:48:39 [MgmtSrvr] INFO     -- The default config
directory '/usr/mysql-cluster' does not exist. Trying to create it...
2018-07-25 21:48:39 [MgmtSrvr] INFO     -- Successfully created config
directory
```

Installation des nœuds de données

- Pour installer les fichiers binaires des nœuds de données, nous devons d'abord récupérer le fichier d'installation .deb approprié à partir de MySQL officiel.

```
wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-
7.6/mysql-cluster-community-data-node_7.6.6-
1ubuntu18.04_amd64.deb
sudo apt install libclass-methodmaker-perl
sudo dpkg -i mysql-cluster-community-data-node_7.6.6-
1ubuntu18.04_amd64.deb
```

Configuration du nœud de données

- Les nœuds de données prennent leur configuration de l'emplacement standard de MySQL, /etc/my.cnf.
- Ajoutez le paramètre de configuration suivant
- La spécification de l'emplacement du nœud de Management est la seule configuration nécessaire au démarrage de ndbd. Le reste de la configuration sera extrait directement du gestionnaire.

```
[mysql_cluster]
# emplacement du manager
ndb-connectstring=198.51.100.2
```

Démarrage des serveurs de données

- Le nœud de données découvrira que son répertoire de données est /usr/local/mysql/data, conformément à la configuration du gestionnaire. Avant de démarrer le démon, nous allons créer ce répertoire sur le nœud.

```
sudo mkdir -p /usr/local/mysql/data
sudo ndbd
Sortie
2018-07-18 19:48:21 [ndbd] INFO     --  Connected to
'198.51.100.2:1186'
2018-07-18 19:48:21 [ndbd] INFO     --  Allocated nodeid: 2
```

Configuration et démarrage du serveur et du client MySQL

- **Un serveur MySQL standard, tel que celui disponible dans le référentiel APT d'Ubuntu, ne prend pas en charge le NDB du moteur MySQL Cluster**.
- Il faut donc le prendre via le site officiel

```
sudo apt install libaio1 libmecab2
wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster_7.6.6-1ubuntu18.04_amd64.deb-bundle.tar
mkdir install
tar -xvf mysql-cluster_7.6.6-1ubuntu18.04_amd64.deb-bundle.tar -C install/
```

Configuration du serveur MySQL

- La configuration de MySQL Server est stockée dans le fichier /etc/mysql/my.cnf par défaut.
- Ajouter la configuration pour signifier:
  - L'utilisateur du storage engine nbd
  - L'emplacement du serveur de management

```
[mysqld]
# Options for mysqld process:
ndbcluster                          # run NDB storage engine

[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=198.51.100.2  # location of management server
```

Vérification de l'installation

- Connecter vous au server SQL
- Verifier l'état le statut de NDB

```
mysql -u root -p
SHOW ENGINE NDB STATUS \G
*************************** 1. row ***************************
  Type: ndbcluster
  Name: connection
Status: cluster_node_id=4, connected_host=198.51.100.2, connected_port=1186,
number_of_data_nodes=2, number_of_ready_data_nodes=2, connect_count=0
```

Vérification de l'installation

Notez ici le nombre de ready_data_nodes: 2.

- Cette redondance permet à votre cluster MySQL de continuer à fonctionner même en cas de défaillance d'un des nœuds de données. Cela signifie également que la charge de vos requêtes SQL sera équilibrée entre les deux nœuds de données.
- Vous pouvez essayer d'arrêter l'un des nœuds de données pour tester la stabilité du cluster. Vous devriez voir la valeur de number_of_ready_data_nodes passer à 1 et revenir à 2.
- Ce test permet de vérifier que :
    - Le serveur SQL fonctionne
    - Il se connecte au serveur de management et récupère les informations

Vérification depuis le manager de données

- Ouvrez la console de gestion de cluster, ndb_mgm.
- Une fois à l'intérieur de la console, entrez la commande SHOW

```
ndb_mgm
ndb_mgm > SHOW

Connected to Management Server at: 198.51.100.2:1186
Cluster Configuration
---------------------
[ndbd(NDB)] 2 node(s)
id=2    @198.51.100.0  (mysql-5.7.22 ndb-7.6.6, Nodegroup: 0, *)
id=3    @198.51.100.1  (mysql-5.7.22 ndb-7.6.6, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @198.51.100.2  (mysql-5.7.22 ndb-7.6.6)

[mysqld(API)]   1 node(s)
id=4    @198.51.100.2  (mysql-5.7.22 ndb-7.6.6)
```

Vérification depuis le manager de données

- Ce qui précède montre qu'il existe deux nœuds de données connectés avec les ID de nœud 2 et 3. Il existe également un nœud de gestion avec l'ID de nœud 1 et un serveur MySQL avec l'ID de nœud 4. Vous pouvez afficher davantage d'informations sur chaque ID en tapant son numéro avec la commande STATUS

```
ndb_mgm > 2 STATUS
Node 2: started (mysql-5.7.22 ndb-7.6.6)
```

Test du cluster

- Pour illustrer les fonctionnalités du cluster, créons une nouvelle table à l'aide du moteur NDB et insérons-y quelques exemples de données.
- Tout d'abord, créons une base de données appelée clustertest avec la commande:
- Maintenant, créez une table simple appelée test_table
- Lorsque vous insérez des données dans une table ndbcluster et que vous sélectionnez des données à partir de cette table, la charge du cluster équilibre les requêtes entre tous les nœuds de données disponibles. Cela améliore la stabilité et les performances de l'installation de votre base de données MySQL.

```
mysql> CREATE DATABASE clustertest;
mysql> CREATE TABLE test_table (name VARCHAR(20), value
VARCHAR(20)) ENGINE=ndbcluster;
mysql> INSERT INTO test_table (name,value)
VALUES('some_name','some_value');
```

Sizing de MySQLCluster

- Qu'elles sont les principaux paramètre de MySQL Cluster

NDB Paramètre

L'idée est de présenter les principales variables possible pour un cluster MySQL:

- NoOfReplicas : L'utilisation de 2 réplicas est recommandée pour garantir la disponibilité des données. Utiliser seulement 1 réplica ne fournit aucune redondance, ce qui signifie que la défaillance d'un seul nœud de données entraîne la totalité du cluster. Les tests sont mal fait pour un nombre de replicas supérieur à 2
- DataDir : Emplacement physique des données NDB
- LockPagesInMainMemory: il est possible de verrouiller un processus en mémoire et d'éviter ainsi tout basculement sur le disque. Ceci peut être utilisé pour garantir les caractéristiques en temps réel du cluster.
  - Ce paramètre prend l'une des valeurs entières :
    - 0: désactive le verrouillage. Ceci est la valeur par défaut.
    - 1: effectue le verrouillage après l'allocation de mémoire pour le processus.
    - 2: effectue le verrouillage avant que la mémoire du processus ne soit allouée.
  - La valeur de 1 évite un swap du process et améliore la réactivité du cluster

NDB Paramètre

- Odirect : Si vous activez ce paramètre, NDBCLUSTER essaie d'utiliser O_DIRECT pour l'écriture des log, afin de réduire la charge sur CPU.
- LockExecuteThreadToCPU : Assigne le process d'exécution à un CPU dédié.
- LockMaintThreadsToCPU : Assigne le process de maintenance à un cpu dédié.
- DataMemory : Mémoire associé aux tuples et aux index
- IndexMemory : Mémoire associé au clef unique et primary key (généralement 20% du DataMemory)

NDB Parametre

- MaxNoOfConcurrentTransactions : Chaque nœud de données du cluster nécessite un enregistrement de transaction pour chaque transaction active du cluster. La tâche de coordination des transactions est répartie sur tous les nœuds de données. Le nombre total d'enregistrements de transaction dans le cluster correspond au nombre de transactions d'un nœud donné multiplié par le nombre de nœuds du cluster.
  - TotalNoOfConcurrentTransactions = (nombre de tables consultées dans une transaction + 1) * nombre de nœuds SQL
  - De plus, chaque transaction implique au moins une opération. pour cette raison, la valeur définie pour MaxNoOfConcurrentTransactions ne doit toujours pas être supérieure à la valeur de MaxNoOfConcurrentOperations.

NDB Parametre

- MaxNoOfConcurrentOperations :Ce paramètre doit être défini au minimum sur le nombre d'enregistrements à mettre à jour simultanément dans les transactions, divisé par le nombre de nœuds de données du cluster. Par exemple, dans un cluster comportant quatre nœuds de données et censé gérer un million de mises à jour simultanées à l'aide de transactions, vous devez définir cette valeur sur 1000000/4 = 250000.

NDB Parametre

- MaxNoOfTables, MaxNoOfUniqueHashIndexes,MaxNoOfOrderedIndexes : Nombre maximal de table, d'index de type hash et d'index order pouvant être ouvert.
- TimeBetweenGlobalCheckpoints, TimeBetweenLocalCheckpoints : temps entres les différents globaux et locaux checkpoints. Lorsqu'une transaction est validée, elle est validée en mémoire principale dans tous les nœuds sur lesquels les données sont mises en miroir. Une transaction est placée dans un groupe de points de contrôle global. Ensuite, les enregistrements du journal de ce groupe sont vidés sur le disque, puis l'ensemble du groupe de transactions est validé sur le disque en toute sécurité sur tous les ordinateurs du cluster.

Partitionnement des tables et répartition des données sur les nœuds du cluster.

- Comment sont placées les données dans le cluster.
- Compréhension du module interne ndb

Architecture

Le clustering MySQL est composé de plusieurs couches due à la complexité de la mise en place d'un shared nothing serveur.

La première couche est la couche applicative qui se connecte au(x) serveur(s) SQL de façon normal sans avoir a modifier le client.

La deuxième couche est la couche SQL où résident les serveurs MySQL.

Le nombre de serveurs MySQL est totalement indépendant du nombre de nœuds de données, ce qui permet une grande flexibilité.

dans la configuration d'un cluster. Pour information le nombre de noeuds SQL peut être augmenté sans arrêter le cluster.

Architecture

- Outre les serveurs MySQL, d'autres programmes communiquant directement avec les nœuds de données (tels que
- programmes de restauration ou programmes permettant d'afficher les tables de cluster).
- MySQL Cluster offre une API C facilite la création de programmes qui communiquent directement avec les nœuds de données sans passer via un serveur MySQL
- La couche de données désignent les nœuds de données. Ils gèrent toutes les données, index et transactions de la cluster et sont donc responsables de la disponibilité et de la durabilité des données.

Partition

Chaque table du cluster est partitionnée en fonction du nombre de nœuds de données. Cela signifie que si le cluster a deux nœuds de données, chaque table est partitionnée en deux parties. Si le cluster comporte quatre nœuds de données, chaque table est partitionnée en quatre parties. Le partitionnement est effectué par défaut en fonction de la valeur de hachage de la clé primaire.

La fonction de partitionnement peut être modifiée si nécessaire, mais dans la plupart des cas, la valeur par défaut est suffisante.

Chaque nœud de données contient la réplique primaire ou le fragment d'une partition, ce qui permet une distribution uniforme des données entre tous les nœuds de données. Pour assurer la disponibilité (et la redondance) des données, chaque nœud contient également une copie d'une autre partition, appelée réplique secondaire.

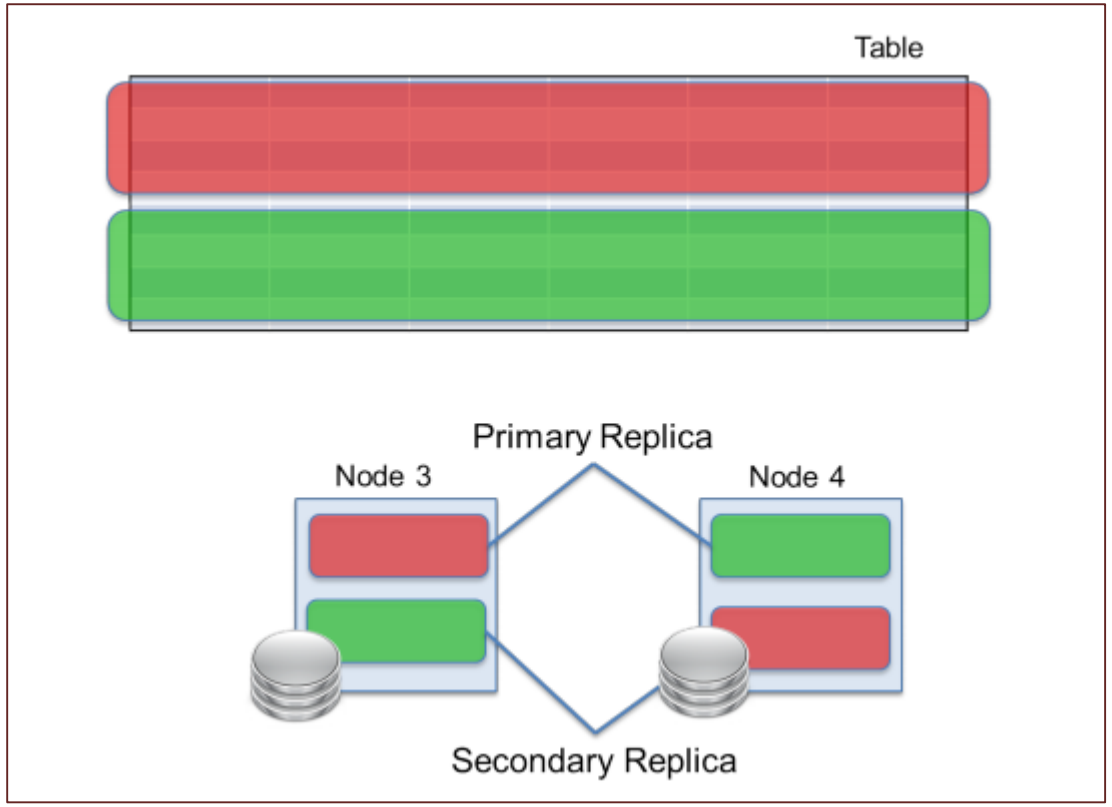
Partition

Les nœuds fonctionnent par paires de sorte que le nœud qui héberge la partition secondaire de la partition principale d'un autre nœud produise un échange réciproque et donne sa propre partition principale en tant que partition secondaire au même partenaire de nœud.

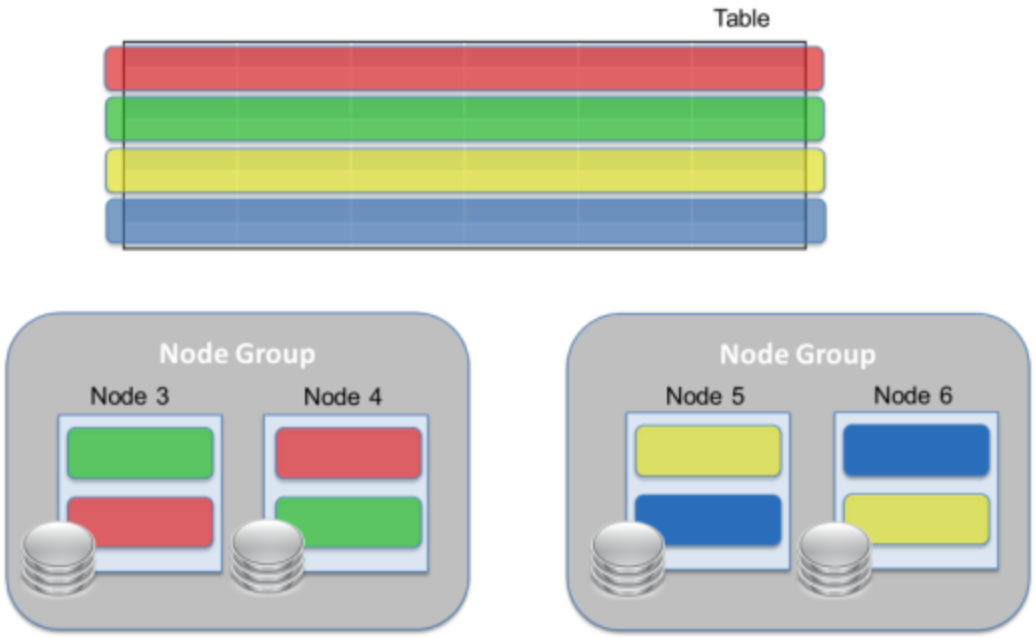
Ces paires sont appelées groupes de noeuds et il y aura noeuds / 2 groupes du nombre de noeuds dans le cluster.

Cela signifie que pour un cluster avec 2 nœuds de données, chaque nœud contiendra la base de données complète, mais pour un cluster avec 4 nœuds, chaque nœud ne contiendra que la moitié des données.

Cluster avec deux noeuds



Cluster 4 noeuds



Détection de vie

En raison de l'architecture sans partage, tous les nœuds du cluster doivent toujours avoir la même vue sur les personnes connectées au cluster. La détection des nœuds en panne est extrêmement importante.

Dans MySQL Cluster, cela peut être géré via TCP close ou via HeartBeat.

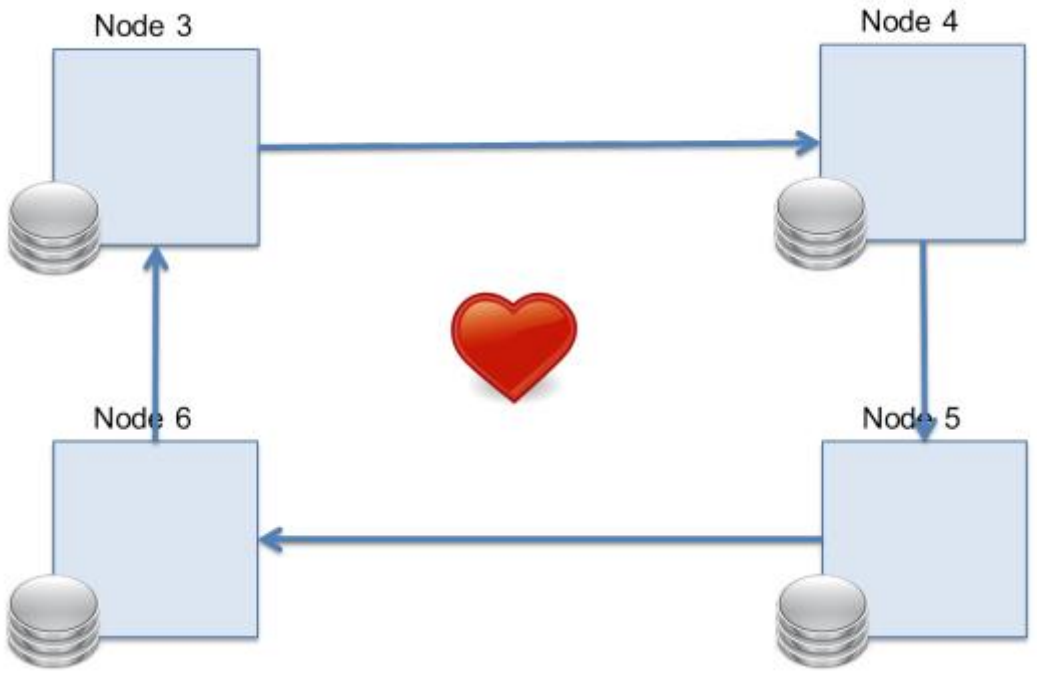
Les noeuds de données sont organisés en un cercle logique, chaque noeud envoyant des pulsations au noeud suivant du cercle.

Détection de vie

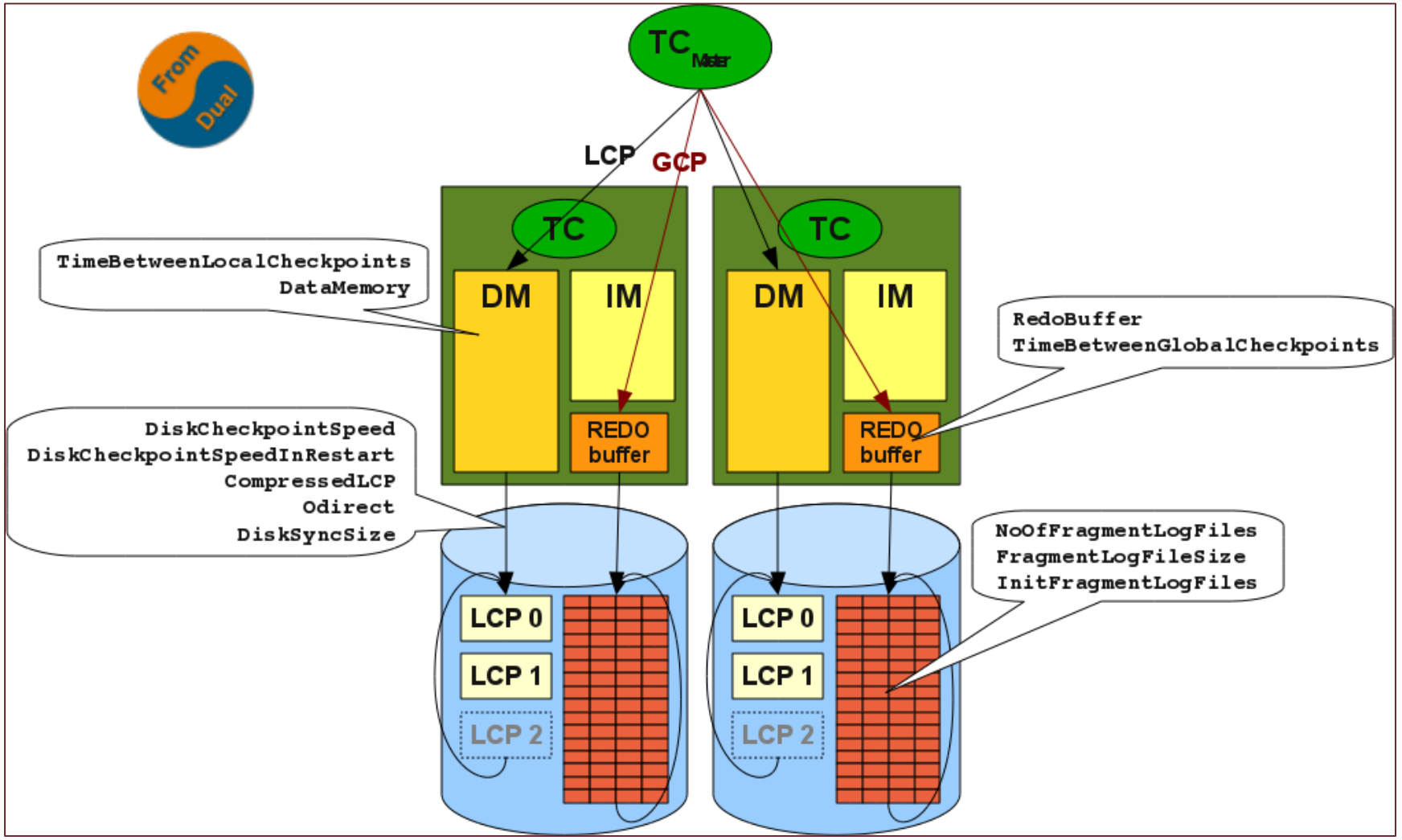
Si un nœud ne parvient pas à envoyer 3 pulsations consécutives, le nœud suivant suppose que le nœud précédent est tombé en panne ou ne peut pas communiquer et lance le protocole de partitionnement du réseau. Pendant ce temps, tous les traitements du cluster sont temporairement arrêtés et les noeuds restants déterminent s'ils peuvent ou non continuer. Le nœud qui ne répond pas est exclu du cluster et les nœuds restants forment un «nouveau» cluster sans le nœud qui ne répond pas.

Notez que dans ce nouveau cluster, le partenaire du nœud qui ne répond pas contiendra désormais deux fragments principaux, car le fragment qui était auparavant un réplica secondaire est maintenant promu au statut principal. Cela signifie également que ce nœud traitera deux fois plus de trafic qu'avant le crash. Tous les nœuds doivent donc être réglés pour pouvoir gérer deux fois la charge de travail normale.

Détection de vie



Resistance à la panne



Résistance à la panne

Étant donné que le cluster MySQL est généralement une base de données en mémoire, il est nécessaire de prendre des mesures pour pouvoir arrêter le cluster sans perte de données. Les nœuds de données utilisent deux procédures pour gérer les données sur le disque. Le premier est le journal REDO. Lorsqu'une transaction a lieu sur un nœud, elle est stockée dans un tampon de journal REDO qui est vidé de manière synchrone sur le disque à intervalles Le journal REDO sur disque ne contiendra que les transactions qui ont été validées au moment où le vidage a eu lieu. L'ecriture a lieu via un point de contrôle global ou GCP.

Résistance à la panne

Bien sûr, un tel journal REDO continuerait de croître à l'infini, le processus doit donc être limité. Cela se fait par un autre processus appelé point de contrôle local ou LCP. Au cours d'un LCP, les nœuds de données stockent un instantané de toutes leurs données sur le disque, ce qui permet de supprimer tout le contenu du journal REDO avant ce moment. Pour des raisons de sécurité, les nœuds de données conservent deux LCP sur le disque et ont donc besoin des journaux REDO à compter du début du premier LCP jusqu'au début du troisième LCP pour qu'une récupération aboutisse.

Résistance à la panne

Les LCP sont écrits de manière circulaire (le LCP 3 remplacera le LCP 1, etc.), de même que le journal REDO. Le contenu des LCP et du journal REDO est utilisé lors de la récupération d'un nœud. Il peut reconstruire les données jusqu'à son dernier GCP à partir de son propre disque, puis transférer les modifications restantes de son nœud partenaire avant de retrouver le statut principal de ses fragments. Lorsque l'ensemble du cluster est arrêté correctement, un dernier GCP est émis avant l'arrêt des noeuds de données. Chaque nœud de données peut alors récupérer ses données complètement à partir de son propre disque.

Transaction

MySQL Cluster assure une réplication synchrone entre les nœuds de données, ce qui signifie que la réplication est synchrone du point de vue du client. Pour ce faire, un algorithme complexe appelé protocole de validation à deux phases (2PC) est utilisé.

Dans le 2PC, une transaction est validée en deux phases, une phase de préparation et une phase de validation.

- Pendant la phase de préparation, les nœuds effectuent les opérations demandées et sont prêts à valider la transaction.
- Pendant la phase de validation, la transaction est validée et les modifications ne peuvent plus être annulées.

Transaction

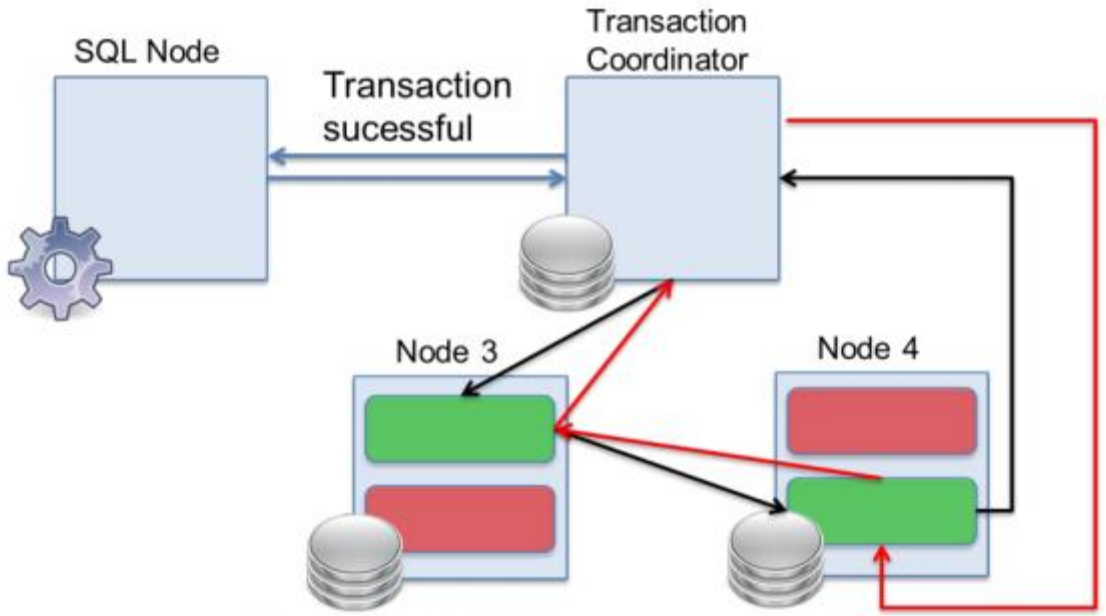


Illustration. 5: Two-phase commit protocol

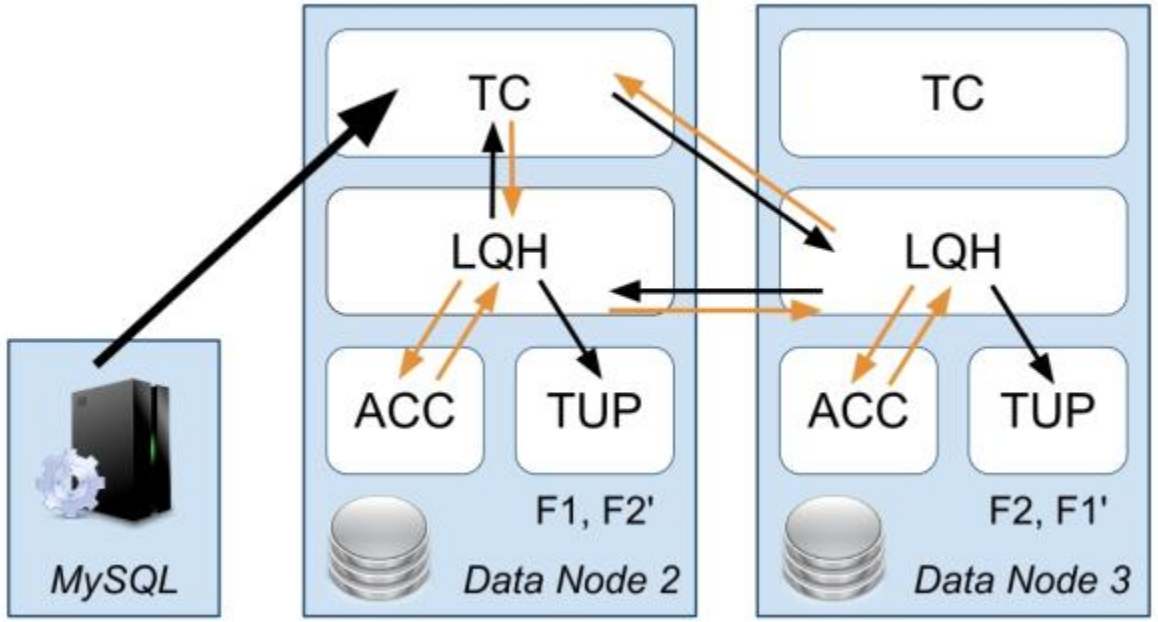
Transaction

Tout d'abord, le serveur MySQL (ou le nœud SQL d'où provient la requête) contacte un nœud de données. Chaque nœud de données possède un module de coordinateur de transaction (TC) et le nœud de données contacté devient le coordinateur ou le gestionnaire de cette transaction. Compte tenu de la valeur de clé primaire, le TC calcule dans quelle partition la ligne réside et transmet l'opération au nœud de données contenant le réplica principal de cette partition. Sur le noeud principal, l'opération est effectuée et la ligne requise est verrouillée, puis la même opération est envoyée au noeud partenaire qui détient le réplica secondaire. Une fois que le nœud secondaire a effectué l'opération, le TC est à nouveau contacté et l'opération peut maintenant être validée.

Transaction

La phase de validation suit la même séquence mais dans l'ordre inverse. le nœud avec le réplica secondaire est contacté en premier, puis le nœud avec le réplica principal. Une fois que le nœud principal a validé l'opération, le TC reconnaît l'accomplissement de l'opération au client. Au total, il s'agit d'un total de 6 messages internes envoyés entre les nœuds et de 2 messages entre le client et le TC pour cette opération de clé primaire simple. Si une transaction contient plusieurs opérations, les étapes internes sont répétées pour chaque opération. Les opérations autres que les écritures de clé primaire suivent des flux de travail quelque peu différents mais le principe reste le même

Transaction



Sauvegardes, restaurations.

- Comment faire un backup de MySQL en mode cluster

Sauvegarde d'un cluster

Du fait de la nature dual d'un cluster MySQL (serveur SQL et cluster), il y a deux méthodes de sauvegarde du cluster:

- La méthode logique via mysqldump
- La méthode propre à MySQLCluster

Méthode NDB

Les données et métadonnées détenues par les nœuds de données d'un cluster NDB actif peuvent être sauvegardées à l'aide de la fonctionnalité de sauvegarde native de NDB. Il s'agit d'une méthode simple et stricte pouvant être lancée à partir de n'importe quel membre du cluster avec le client de gestion.

/usr/bin/ndb_mgm -e "START BACKUP <id>“

où ‘id’ est un entier qui identifie l’instantané à créer. Il doit être spécifié pour les exécutions non interactives à l’aide du paramètre -e.

Méthode NDB

Lorsque cette commande est émise, chaque démon ndbd sur les nœuds de données crée un instantané de son segment de base de données stocké localement. La capture instantanée est stockée dans le système de fichiers local, sous le chemin du répertoire de sauvegarde configuré sur le noeud de gestion (le chemin par défaut est / usr / local / mysql / data / BACKUP). Un répertoire est créé pour chaque instantané afin de contenir ses fichiers de vidage binaire portant le nom «BACKUP».

Methode Logique

- Cette méthode est la méthode de la version GPL.
- Basiquement il s'agit d'un script appellé pour toutes les bases de données contenant au moins une table avec un moteur non-ndbcluster et non-mémoire.

```
[client]
user = backup
password =

[mysql]
batch
skip-column-names

[mysqldump]
single-transaction
quote-names
routines
triggers
events
force
```

Restoration des données MySQL Cluster

- La méthode logique consiste a inserer un script de SQL dans le nœud SQL.
- Il s'agit d'une restauration normal

```
/usr/bin/mysql -u <user_name> -p <db_name> < <dump_file>
```

ndb_restore

- Lors de la récupération des données à partir de ces sauvegardes, ndb_restore doit être exécuté une fois pour chaque fichier de sauvegarde, autrement dit, une fois pour chaque nœud de données du cluster au moment de la sauvegarde. Pour éviter les incohérences, il est recommandé d'activer le mode mono-utilisateur sur le cluster avant de commencer, ce qui garantit qu'au moment de la récupération, seul le processus ndb_backup a accès aux nœuds de données.

```
/usr/bin/ndb_mgm -e "ENTER SINGLE USER MODE <ndb_node_id>"
```

ndb_restore

```
/usr/bin/ndb_restore \
[-c ] \
-n \
-b \
--backup_path= \
--ndb-nodeid= \
[--include-tables=] \
[--include-databases=] \
[--exclude-tables=] \
[--exclude-databases=] \
-m
```

- connection_string est l'adresse IP ou le nom d'hôte du ndb_mgmd auquel se connecter. Cette option peut être omise lors de l'exécution de la commande sur le noeud de gestion.
- data_node_id est l'ID du noeud de données sur lequel la restauration doit être effectuée
- ibackup_id est l'identifiant numérique de l'instantané à partir duquel les fichiers doivent être récupérés.
- backup_path est le chemin du répertoire de sauvegarde. Il doit s'agir du répertoire dans lequel les fichiers de sauvegarde ont été restaurés à l'étape 1.
- ndb-nodeid est le même node_id qui a été mentionné ci-dessus lors de la discussion sur le mode mono-utilisateur.
- table_list, db_list contient le nom des tables et / ou bases de données (délimitées par des virgules), qui doivent ou non être restaurés à partir de dump:
  - quand rien n'est donné, tout est restauré;
  - quand seulement exclure est donné, tous les objets sauf les objets nommés sont restaurés;
  - quand seulement include est donné, seuls les objets nommés sont restaurés

Sortie du mode single user mode

- Pour quitter le mode mono-utilisateur après la récupération, exécutez la commande suivante.
- En cas de problème, le redémarrage des nœuds de données les ramènera également en mode normal.

```
/usr/bin/ndb_mgm -e "EXIT SINGLE USER MODE"
```

MySQL Cluster Limitation

Non-conformité à la syntaxe SQL

- Tables temporaires. Les tables temporaires ne sont pas prises en charge.
- Les clés et les index sur les tables de cluster NDB sont soumis aux limitations suivantes:
  - Largeur de colonne. Si vous tentez de créer un index sur une colonne de table NDB dont la largeur est supérieure à 3072 octets, vous réussissez, mais seuls les 3072 premiers octets sont réellement utilisés pour l'index.
  - Colonnes TEXT et BLOB. Vous ne pouvez pas créer d'index sur les colonnes de table NDB qui utilisent les types de données TEXT ou BLOB.
  - Index FULLTEXT. Le moteur de stockage NDB ne prend pas en charge les index FULLTEXT
- HASH Key and NULL. L'utilisation de colonnes nullables dans les clés uniques et les clés primaires signifie que les requêtes utilisant ces colonnes sont traitées comme des analyses de table complètes.
- BIT. Une colonne BIT ne peut pas être une clé primaire, une clé unique ou un index, ni faire partie d'une clé primaire composite, d'une clé unique ou d'un index.
- Colonnes AUTO_INCREMENT :Dans le cas d'une table NDB sans clé primaire explicite, une colonne AUTO_INCREMENT est automatiquement définie et utilisée en tant que clé primaire «masquée». Pour cette raison, vous ne pouvez pas définir une table comportant une colonne AUTO_INCREMENT explicite à moins que cette colonne soit également déclarée à l'aide de l'option PRIMARY KEY.

Non-conformité à la syntaxe SQL

Restrictions sur les clés étrangères. La prise en charge des contraintes de clé étrangère dans NDB 7.5 est comparable à celle fournie par InnoDB, sous réserve des restrictions suivantes:

- Chaque colonne référencée en tant que clé étrangère nécessite une clé unique explicite, si ce n'est pas la clé primaire de la table.

- ON UPDATE CASCADE n'est pas pris en charge lorsque la référence concerne la clé primaire de la table parent.

Limitation au regard du serveur MySQL

- Utilisation de la mémoire et récupération. La mémoire consommée lorsque les données sont insérées dans une table NDB n'est pas automatiquement récupérée lors de la suppression. Une instruction DELETE sur une table NDB rend la mémoire précédemment utilisée par les lignes supprimées disponible pour être réutilisée par des insertions sur la même table uniquement. Cependant, cette mémoire peut être rendue disponible pour une réutilisation générale en exécutant OPTIMIZE TABLE.

- Le nombre maximal de nœuds de données est 48.

- Un nœud de données doit avoir un ID de nœud compris entre 1 et 48 inclus. (Les nœuds de gestion et d'API peuvent utiliser des ID de nœud compris entre 1 et 255, inclusivement.)

- Le nombre maximal de nœuds dans un cluster NDB est de 255. Ce nombre inclut tous les nœuds SQL (serveurs MySQL), les nœuds API (applications accédant au cluster autres que les serveurs MySQL), les nœuds de données et les serveurs de gestion.

- Le nombre maximal d'objets de métadonnées dans les versions actuelles du cluster NDB est de 20320. Cette limite est codée en dur.

Limitation sur les transactions

- Le moteur de stockage NDBCLUSTER prend uniquement en charge le niveau d'isolation de transaction READ COMMITTED.
- BLOB ,TEXT. NDBCLUSTER stocke uniquement une partie dela colonne dans la table. Le reste du BLOB ou du TEXT est stocké dans une table interne séparée non accessible à MySQL.
- Donc si le SELECT inclut une colonne BLOB ou TEXT, le niveau d'isolement de la transaction READ COMMITTED est converti en lecture avec verrou de lecture. Ceci est fait pour garantir la cohérence

Limitation sur les transactions

- Un cluster NDB ne traite pas correctement les transactions volumineuses; il est préférable d'effectuer un certain nombre de petites transactions avec quelques opérations chacune plutôt que de tenter une seule transaction importante contenant un grand nombre d'opérations.
- Les transactions volumineuses nécessitent de très grandes quantités de mémoire.
- LOAD DATA INFILE n'est pas transactionnel lorsqu'il est utilisé sur des tables NDB.
- Lors de la copie d'une table NDB dans le cadre d'une table ALTER TABLE, la création de la copie n'est pas transactionnelle.

Limitations sur les objets

- Noms de bases de données et de tables. Lors de l'utilisation du moteur de stockage NDB, la longueur maximale autorisée pour les noms de base de données et pour les noms de table est de 63 caractères. Une instruction utilisant un nom de base de données ou un nom de table supérieur à cette limite échoue avec une erreur appropriée.

- Nombre d'objets de base de données. Le nombre maximal d'objets de base de données NDB dans un même cluster NDB (y compris les bases de données, les tables et les index) est limité à 20320.

- Attributs par table. Le nombre maximal d'attributs (c'est-à-dire de colonnes et d'index) pouvant appartenir à une table donnée est de 512.

- Attributs par clé. Le nombre maximal d'attributs par clé est de 32.

- Taille de la ligne. La taille maximale autorisée d'une ligne est de 14 000 octets. Chaque colonne BLOB ou TEXT contribue à ce total avec 256 + 8 = 264 octets.

- Stockage de colonne BIT par table. La largeur combinée maximale de toutes les colonnes de bit à bits utilisées dans un tableau NDB donné est de 4096.