Optimiser une base de données MySQL

Mr.CHAOULID

1. Optimiser les requêtes SQL :

Quelques conseils pour optimiser les requêtes SQL:

- Éviter si possible les SELECT * et réduire le nombre de champs, afin de réduire les données chargées en mémoire ;
- Éviter d'utiliser des fonctions dans les prédicats : exemple SELECT * FROM TABLE1 WHERE UPPER(COL1)='ABC‘ ;
- Éviter d'utiliser un caractère générique (%) au début d'un prédicat like ;
- Utiliser la jointure interne (inner join), au lieu de la jointure externe (outer join) si possible ;
- N'utiliser DISTINCT et UNION que si nécessaire ;
- N'utiliser ORDER BY que si nécessaire ;
- Utiliser les requêtes préparées ou les procédures stockées facilite la mise en cache des requêtes en interne par MySQL et assure un bon niveau de sécurité ;
- Utiliser la clause EXPLAIN pour comprendre le fonctionnement d’une requête et quelles sont les clauses qui impactent ses performances

2. Les requêtes préparées:

- Une requête préparée en MySQL est une manière de gérer des requêtes SQL de manière sécurisée et optimisée. Les instructions préparées sont principalement utilisées pour exécuter des requêtes SQL répétitives avec des paramètres variables, ce qui peut améliorer les performances et renforcer la sécurité de votre application.
- **Exemple:**

```
PREPARE MaReq FROM 'SELECT * FROM EMPLOYE WHERE MATRICULE=?';
SET @mat = 1;
EXECUTE MaReq USING @mat;
DEALLOCATE PREPARE MaReq;
```

- Elles sont plus efficaces pour les requêtes répétitives car le moteur de base de données peut les compiler une fois et les réutiliser avec différents paramètres.

3. Les index:

- L'index en MySQL est un mécanisme qui améliore les performances de recherche dans une table de base de données. Un index fonctionne de manière similaire à un index dans un livre, car il permet d'accéder rapidement à des données spécifiques au lieu de parcourir l'intégralité de la table. L'utilisation d'index est essentielle pour accélérer les opérations de recherche, de tri et de jointure sur de grandes tables.
- En d'autres termes, un index crée un moyen efficace de localiser rapidement des données en évitant de parcourir l'intégralité de la table.
- Un index est une structure de données qui répertorie les valeurs d'une ou plusieurs colonnes spécifiques de la table et pointe vers les emplacements physiques des enregistrements correspondants.
- **Syntaxe:**

```
CREATE INDEX nom_de_l_index ON nom_de_la_table (nom_de_la_colonne);
```

- **Exemple:**

```
CREATE INDEX INDEX_VILLE ON EMPLOYE(VILLE);
```

3. Les index:

- Vous pouvez supprimer un index à l'aide de la commande **DROP INDEX**
- **Syntaxe:**

```
DROP INDEX nom_de_l_index ON nom_de_la_table;
```

- **Exemple:**

```
DROP INDEX INDEX_VILLE ON EMPLOYE;
```

- **Remarque:**

Trop d'index sur une table peut ralentir les opérations de mise à jour (INSERT, UPDATE, DELETE) car les index doivent être mis à jour en conséquence. Il est essentiel de choisir judicieusement les colonnes à indexer pour éviter une surutilisation.

4. Optimiser la configuration de serveur MySQL.

Quelques conseils pour l'optimisation des performances d’un serveur MySQL :

- Si on utilise des disques durs traditionnels (HDD), on peut effectuer une mise à niveau vers des disques à semiconducteurs (SSD) pour une amélioration des performances.
- On peut ajuster la cache mémoire pour améliorer les performances. Si on n'a pas assez de mémoire ou si la mémoire existante n'est pas optimisée, on peut finir par nuire à nos performances au lieu de les améliorer.
- MyISAM est un ancien style de base de données utilisé pour certaines bases de données MySQL. C'est une conception de base de données moins efficace. Le plus récent InnoDB prend en charge des fonctionnalités plus avancées et dispose de mécanismes d'optimisation intégrés.
- Essayer d'utiliser la dernière version de MySQL
- Envisager d'utiliser un outil d'amélioration automatique des performances(tuning-primer, MySQLTuner,..)

FIN