# Améliorer les performances grâce aux INDEX

|  |  |
| --- | --- |
| Objectifs | Savoir créer des index simples, composites, unique, etc |
| Méthodologie | Travail individuel |
| Durée estimée | 20 min (avec le corrigé) |
| Type de base de données : | MySQL |

**Exercice 1** :

Voici un exercice simple pour comprendre l'importance des index en MySQL.

**Étape 1**: **Création de la table et insertion de données**

Tout d'abord, créez une base de données **db\_index** ainsi qu’une table `Etudiants` avec les colonnes `id`, `nom`, et `prenom`.

|  |
| --- |
| **CREATE** **TABLE** Etudiants (  id **INT PRIMARY KEY AUTO\_INCREMENT**,  nom **VARCHAR**(50),  prenom **VARCHAR**(50),  pseudo **VARCHAR**(50),  **UNIQUE** (pseudo)  ); |

Ensuite, insérez quelques données dans cette table :

|  |
| --- |
| **INSERT** **INTO** Etudiants (nom, prenom, speudo) **VALUES**  ('Doe', 'John', 'doeJohn'),  ('Smith', 'Jane', 'smiJane'),  ('Brown', 'Emily', 'broEmily'),  ('Williams', 'Michael', 'wilMichael'),  ('Johnson', 'Sarah', 'johSarah'); |

**Étape 2** : **Requête sans index**

Essayez d'exécuter la requête suivante pour rechercher un étudiant par son nom :

|  |
| --- |
| **SELECT** \* **FROM** Etudiants **WHERE** nom = 'Brown'; |

À ce stade, la colonne `nom` n'a pas d'index, donc MySQL doit effectuer une recherche séquentielle, ce qui peut être lent sur une grande table.

**Étape 3 : Examiner le plan de requête**

Utilisez la commande `EXPLAIN` pour voir si un index est utilisé :

|  |
| --- |
| **EXPLAIN** **SELECT** \* **FROM** Etudiants **WHERE** nom = 'Brown'; |

Normalement aucun index est utilisé

**Étape 4 : Ajouter un index**

Ajoutez un index sur la colonne `nom` :

|  |
| --- |
| **CREATE** **INDEX** idx\_nom **ON** Etudiants (nom); |

**Étape 5 : Requête avec index**

Exécutez à nouveau la requête de recherche par nom :

|  |
| --- |
| **SELECT** \* **FROM** Etudiants **WHERE** nom = 'Brown'; |

Le temps d'exécution devrait être le même ou plus rapide, surtout si la table avait beaucoup plus de lignes. Le vrai gain de performance serait plus visible sur des tables beaucoup plus grandes.

**Étape 6 : Examiner le plan de requête**

Utilisez la commande `EXPLAIN` pour voir comment l'index est utilisé :

|  |
| --- |
| **EXPLAIN** **SELECT** \* **FROM** Etudiants **WHERE** nom = 'Brown'; |

Vous devriez voir que MySQL utilise l'index `idx\_nom` pour cette requête, ce qui est généralement plus rapide que de scanner toute la table.

**Étape 7 :**

Soit la requête suivante :

|  |
| --- |
| **SELECT** \* **FROM** Etudiants **WHERE** pseudo = 'smiJane'; |

**Travail à réaliser** :

* Pour optimiser cette requête, de quel type d’index avez-vous besoin ?
* Quelle est la commande pour créer un tel index ? ATTENTION vous ne devez pas créer l’index
* Utiliser la requête avec le mot clé EXPLAIN sans avoir créé l’index
* Que remarquez-vous ?

**Solution :**

* **Index unique**
* CREATE UNIQUE INDEX idx\_pseudo ON Etudiants (pseudo);
* EXPLAIN SELECT \* FROM Etudiants WHERE pseudo = 'smiJane';
* **On remarque que l’index existe déjà, en effet, c’est** la DB qui la créé

**Exercice 2 :**

Prenons un exemple concret basé sur une table qui contient des informations sur des ventes réalisées dans un magasin.

**Table des ventes :**

Dans la même base de données, créer la table `ventes` suivante :

|  |
| --- |
| **CREATE** **TABLE** ventes (  vente\_id INT PRIMARY KEY,  date\_vente DATE,  vendeur\_id INT,  produit\_id INT,  quantite INT  ); |

Dans cette table :

- `vente\_id` est un identifiant unique pour chaque vente.

- `date\_vente` est la date à laquelle la vente a été réalisée.

- `vendeur\_id` est l'identifiant du vendeur qui a réalisé la vente.

- `produit\_id` est l'identifiant du produit vendu.

- `quantite` est le nombre d'unités du produit vendues.

**Scénario** : La direction veut souvent savoir combien d'unités d'un produit spécifique ont été vendues par un vendeur spécifique sur une date spécifique. Une requête courante pourrait ressembler à :

|  |
| --- |
| **SELECT** SUM(quantite)  **FROM** ventes  **WHERE** date\_vente = '2023-09-20' **AND** vendeur\_id = 5 **AND** produit\_id = 101; |

Dans cet exemple, la requête filtre sur trois colonnes : `date\_vente`, `vendeur\_id` et `produit\_id`.

**Travail à réaliser** :

* Pour optimiser cette requête, de quel type d’index avez-vous besoin ?
* Quelle est la commande ?
* Est-ce que l’ordre des colonnes est important ? (A vous de rechercher sur le web pour trouver la solution)

**Solution : Index composite**

Pour optimiser cette requête, un index composite sur les trois colonnes serait utile. L'index permettrait d'accélérer la recherche des enregistrements correspondants dans la table `ventes`.

**CREATE** **INDEX** idx\_ventes\_date\_vendeur\_produit **ON** ventes(date\_vente, vendeur\_id, produit\_id);

Avec cet index composite, MySQL peut rapidement trouver toutes les ventes pour une date, un vendeur et un produit donnés. Cela accélère la sommation des quantités vendues.

L'ordre des colonnes dans l'index est également important. Dans cet exemple, nous avons commencé par `date\_vente` car nous supposons que le filtrage par date est probablement le plus sélectif. Si le contexte d'utilisation de la base de données montre qu'un autre ordre est plus optimal, l'ordre des colonnes dans l'index devrait être ajusté en conséquence.