Optimisation de tables avec les index

## Les catégories d’index les plus utilisés

### b-tree (balanced - tree) - défaut

Type d’index le plus populaire. Voici un exemple :

On dit que cet index est balancé parce que les feuilles sont toujours au même niveau. Les blocs d’index peuvent cependant ne pas contenir le même nombre de valeurs et peuvent contenir de l’espace inutilisé.



Donc, si on fait par exemple : SELECT \* FROM USERS WHERE AGE = 21, l’index permettra de trouver où est la ligne de la table qui répond à la requête.

CREATE INDEX idx\_city ON USERS (city);

## Les index en général

Il faut noter que les clés primaires et étrangères sont automatiquement indexées. Cependant, une grande partie des requêtes faites par les applications/usagers ne sont pas sur ces clés.

Il faut donc avoir une connaissance de l’application pour savoir où mettre les index.

### Index unique

Les index uniques doivent être créés sur des colonnes dont une même valeur est acceptée une fois au maximum.

Quelques exemples de bons candidats : courriel, numéro d’assurance sociale, nom d’usager

CREATE UNIQUE INDEX idx\_username ON USERS(username);

### Index à colonne simple

À utiliser lorsque l’usager fait une requête en filtrant sur une seule colonne.

Exemple : SELECT \* FROM users WHERE birth\_date > TO\_DATE('1990', 'YYYY');

### Index à colonnes multiples (composite index)

CREATE INDEX idx\_nom ON usagers(nom, prenom);

Extrêmement intéressant lorsqu’une requête se fait sur plusieurs colonnes.

Exemple valide pour l’index (province, ville, rue).

* WHERE province = …
* WHERE province = … AND ville = …
* WHERE province = … AND ville = … AND rue = …

Exemple où l’index ne sera pas utilisé:

* WHERE ville = … AND rue = …
* WHERE ville = …
* WHERE rue = …

La requête suivante n’utiliserait donc pas l’index (province, ville, rue) :

SELECT \* FROM users WHERE ville = ‘montreal’ AND rue = ‘Sanguinet’

### Le « Full text indexing »

Utiliser le "full text indexing" permet d’indexer chaque mot dans un texte. Il est plus long à maintenir par la base de données, mais peut accélérer grandement la performance.

Créer un index « Full text » en MYSQL :

CREATE FULLTEXT INDEX idx\_produits\_desc ON produits2(description);

Requête permettant de trouver le mot « xbox » peu importe où il se trouve dans un texte :

SELECT COUNT(\*) FROM produits2 WHERE MATCH(description) AGAINST('xbox');

Ceci est dramatiquement plus performant que cette requête :

SELECT COUNT(\*) FROM produits2 WHERE description LIKE '%xbox%';