OPTIMISATION D'UNE BASE DE DONNÉES

OBJECTIFS

- tirer le maximum du matériel disponible
- tirer le maximum de l'application qui utilise la base de données
- répondre à l'accroissement du volume des données
- répondre à l'accroissement de la charge : nombre de connexions, nombre d'utilisateurs
- éviter de recoder l'application par manque de performances → trop cher
- satisfaire le client

OPTIMISER LA STRUCTURE DE LA BASE DE DONNÉES

- choisir les bons types de données : facilité de manipulation du type de données, optimisation du stockage
- éliminer les données redondantes
- optimiser les jointures entre les tables
- stocker des données calculées ou faire les calculs par requêtes ?
- ajouter des colonnes intelligentes : auto_increment, valeurs par défaut ...
- coder correctement les requêtes (pas de SELECT * …)

CHOIX DU MOTEUR DE STOCKAGE

- un moteur de stockage est un ensemble d'algorithmes utilisés pour stocker les informations et y accéder au moyen de requêtes SQL
- la plupart des SGBD relationnels proposent un **moteur unique**, créé pour être le plus efficace possible
- à l'inverse, MySQL et MariaDB notamment laissent la possibilité de **choisir** pour chaque table quel moteur on veut utiliser → plusieurs moteurs peuvent **coexister** dans la même base → choix de conception avec avantages et inconvénients

CHOIX DU MOTEUR DE STOCKAGE

Le choix du moteur a beaucoup d'influence sur les **performances** du serveur de bases de données et donc sur le **rendu de l'application** pour les utilisateurs

Il influe notamment sur :

- le type de **verrouillage** des données (enregistrement, bloc, table …) et donc sur leur disponibilité
- le mode de **sauvegarde** et de **restauration**
- **l'optimisation** des tables

CHOIX DU MOTEUR DE STOCKAGE

Pour connaître les moteurs disponibles dans MySQL, il faut taper la commande : SHOW ENGINES ;

La requête SQL a été exécutée avec succès.

```
show engines;
```

Profilage [Éditer en ligne] [Éditer] [Créer le code source PHP] [Actualiser]

Options supplémentaires

Engine	Support	Comment	Transactions	XA	Savepoints
CSV	YES	Stores tables as CSV files	NO	NO	NO
MRG_MyISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary...	NO	NO	NO
Aria	YES	Crash-safe tables with MyISAM heritage. Used for i...	NO	NO	NO
MyISAM	YES	Non-transactional engine with good performance and...	NO	NO	NO
SEQUENCE	YES	Generated tables filled with sequential values	YES	NO	YES
InnoDB	DEFAULT	Supports transactions, row-level locking, foreign ...	YES	YES	YES
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO

TRANSACTION SQL

Certains moteurs gèrent les **transactions**

Une transaction est une opération en plusieurs étapes, validée en bloc à la fin

- exemple type : le virement bancaire entre deux comptes
- on débite le premier compte et on crédite le second
- il faut garantir qu'en cas d'échec d'une étape de la transaction, rien ne sera pris validé : si on a pu débiter le premier compte mais si on ne peut pas créditer le second, alors il faut recréditer le premier compte

DEUX GRANDES FAMILLES DE MOTEURS

Un virement de 50 euros du compte de Jean au compte de Pierre ressemble à ceci :

```
BEGIN;
    UPDATE comptes
        SET balance = balance - 50.00 WHERE nom = 'Jean';
    COMMIT;
    UPDATE comptes
        SET balance = balance + 50.00 WHERE nom = 'Pierre';
    COMMIT;
```

Si on ne peut pas créditer Pierre parce que son compte est bloqué par un incident bancaire par exemple, une commande de type **ROLLBACK** peut être utilisée à la place de **COMMIT**. Toutes les mises à jour réalisées jusque-là sont alors annulées.

DEUX GRANDES FAMILLES DE MOTEURS

```
BEGIN;
    UPDATE comptes
        SET balance = balance - 50.00 WHERE nom = 'Jean';
    SAVEPOINT point-de-sauvegarde-1;
    UPDATE comptes
        SET balance = balance + 50.00  WHERE nom = 'Pierre';
  !!! Erreur ; le compte ne peut pas être crédité, retour en arrière
    ROLLBACK TO point-de-sauvegarde-1;
    UPDATE comptes
        SET balance = balance + 50.00 WHERE nom = 'Jean';
COMMIT;
```

INNODB

- InnoDB est un moteur **transactionnel**
- InnoDB est un moteur **relationnel** : les relations entre les données de plusieurs tables sont **cohérentes.** Si on modifie certaines données, ces changements sont répercutés aux tables liées : ON UPDATE CASCADE, ON DELETE CASCADE
- il gère **l'intégrité référentielle** de la base
- il gère aussi le **verrouillage** des données au niveau de chaque enregistrement, contrairement à MyISAM qui gère le verrouillage au niveau de la table entière
- si on écrit des données sur un enregistrement de la table, seul cet enregistrement est bloqué pour les autres utilisateurs, tous les autres sont accessibles

MYISAM

- MyISAM a longtemps été le moteur par défaut de MySQL
- c'est un moteur **non transactionnel** rapide en écriture et **très rapide** en lecture
- ceci vient notamment du fait qu'il **ne gère pas les relations ni les transactions** et évite donc des contrôles gourmands en ressources
- il perd donc en **sûreté** ce qu'il gagne en **vitesse**
- il permet de créer des index sur les champs de type TEXT ce qui rend les recherches beaucoup plus efficaces qu'avec l'instruction LIKE
- c'est donc un moteur efficace pour les fonctions de recherche avancées dans les textes

Critères	InnoDB	MyISAM
Transactions	Oui	Non
Relations	Oui	Non
Cascade	Oui	Non
Intégrité référentielle	Oui	Non
Verrouillage	Enregistrement	Table
Utilisation des ressources	Correcte	Peu gourmand
Vitesse	Correcte	Maximale
Gestion des textes	Correcte	Optimisée