# MySQL

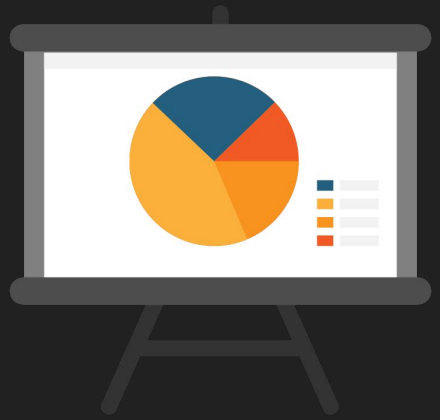
## Structured Query Language


**Rachid EDJEKOUANE (edjek@hotmail.fr)**

# Prérequis

- Connaissance de base en développement web
- Compréhension de base des concepts de bases de données (tables, relations, requêtes, etc.)
- Notions de bases de données
- Ponctualité
- Curiosité

# Introduction



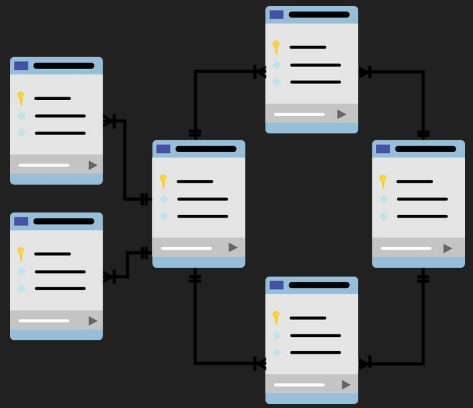
# Base de données

Une base de données est une collection organisée de données structurées, stockées de manière cohérente et accessible électroniquement.

Elle est conçue pour permettre la gestion, la manipulation et la récupération efficaces des informations qu'elle contient.

# Base de données : les tables

Une base de données est généralement composée de tables, qui sont des structures de données rectangulaires organisées en colonnes (champs) et en lignes (enregistrements). Les tables sont reliées entre elles par des relations, ce qui permet d'établir des liens logiques entre les données.



# Base de données : les tables

Une base de données contient une ou plusieurs tables, dont les noms doivent être uniques au sein de la base de la données. Une table contient des colonnes. Les colonnes contiennent les données.

| id | prenom | nom | email | ville |
|---|---|---|---|---|
| 1 | Marine | Leroy | mleroy@example.com | Paris |
| 2 | Jean | René | jrene@example.com | Lyon |
| 3 | Ted | Bundy | tbundy@example.com | Miami |

# SGBDR

SGBDR : Système de Gestion de Base de Données Relationnelle

Fait référence à un logiciel utilisé pour stocker, gérer et interagir avec des données organisées selon le modèle de base de données relationnelle.


ORACLE


PostgreSQL

# MySQL : langage SQL

MySQL utilise le langage SQL (Structured Query Language) pour interagir avec les bases de données.

SQL est un langage standardisé largement utilisé pour la manipulation des données, permettant aux développeurs d'exécuter des requêtes, de créer, de modifier et de supprimer des données dans les tables.

# MySQL : open source

MySQL est distribué sous une licence open source, ce qui signifie qu'il est gratuit et que son code source est accessible à tous.

Cela permet une grande flexibilité et une large communauté de développeurs qui contribuent à son développement et à son amélioration continue.

# MySQL : architecture client-serveur

MySQL suit le modèle client-serveur, où les applications clientes se connectent à un serveur MySQL pour accéder et manipuler les bases de données.

Cela permet une gestion centralisée des données et facilite l'accès concurrentiel aux bases de données.

# Encore plus d’avantages…

Haute performance : MySQL est optimisé pour offrir de bonnes performances, même dans des environnements de traitement de données volumineuses et à forte charge.

Large adoption : MySQL est largement utilisé dans l'industrie, des petites applications web aux grandes entreprises. Il est compatible avec de nombreux langages de programmation et frameworks web, ce qui facilite son intégration dans les projets de développement.

# En résumé

MySQL est un système de gestion de base de données populaire, open source et performant, utilisé par de nombreux développeurs pour stocker, gérer et manipuler les données dans les applications web.

Sa facilité d'utilisation et sa flexibilité en font un choix privilégié pour de nombreux projets de développement.

# ACID

# Principes ACID

Les principes ACID sont un ensemble de propriétés qui garantissent que les transactions de bases de données sont traitées de manière fiable. ACID est un acronyme pour Atomicity, Consistency, Isolation, et Durability.

Ces propriétés assurent que les bases de données restent précises et cohérentes même en cas de défaillances matérielles, logicielles ou humaines.

# Atomicité (Atomicity)

L'atomicité garantit que chaque transaction est traitée comme une unité indivisible.

Cela signifie que toutes les opérations dans une transaction sont exécutées avec succès ou aucune ne l'est. Si une partie de la transaction échoue, la transaction entière échoue, et la base de données est laissée inchangée.

# Cohérence (Consistency)

La cohérence garantit que la base de données passe d'un état valide à un autre état valide après une transaction.

Toutes les règles définies (comme les contraintes, les déclencheurs, etc.) sont respectées avant et après la transaction.

# Isolation (Isolation)

L'isolation garantit que les transactions concurrentes se déroulent de manière isolée et que les opérations d'une transaction ne sont pas visibles aux autres transactions jusqu'à ce qu'elles soient terminées.

Cela prévient les problèmes comme les lectures sales, les lectures non répétables et les lectures fantômes.

# Durabilité (Durability)

La durabilité garantit que les résultats d'une transaction sont permanents et persisteront même en cas de panne système.

Une fois qu'une transaction est validée (committed), ses modifications ne seront pas perdues.

# ACID : Conclusion

Les principes ACID sont cruciaux pour assurer l'intégrité, la fiabilité et la cohérence des transactions dans une base de données. Ils forment la base sur laquelle repose la gestion des transactions dans les systèmes de gestion de bases de données relationnelles (SGBDR), garantissant que les opérations sont exécutées de manière sécurisée et correcte, même en cas de défaillances.

# Découverte

# MySQL : comment ça marche ?

Pour utiliser MySQL, téléchargez sa version sur le site officiel :

https://www.mysql.com/

# MySQL : configuration

- Exécutez le fichier d'installation téléchargé.
- Suivez les instructions de l'assistant d'installation.
- Sélectionnez les composants que vous souhaitez installer (MySQL Server, MySQL Workbench, etc.).
- Choisissez les options de configuration appropriées (par défaut, personnalisées).
- Définissez un mot de passe pour l'utilisateur root (administrateur) de MySQL.
- Terminez l'installation en suivant les instructions.

# En pratique

# MySQL : vérification de l’installation

Voir la version : mysql –version

Après l'installation, vous pouvez vérifier si MySQL fonctionne correctement en ouvrant une ligne de commande (terminal) et en exécutant la commande mysql -u root -p

Vous serez invité à saisir le mot de passe que vous avez défini pendant l'installation.

Si vous êtes en mesure de vous connecter avec succès, cela signifie que MySQL est correctement installé.

# MySQL : créer sa première base de données

Une fois connecté à MySQL, vous pouvez créer une nouvelle base de données en utilisant la commande CREATE DATABASE.

Par exemple, pour créer une base de données appelée "mydatabase", exécutez la commande suivante :

```
CREATE DATABASE my_database;
```

# MySQL : créer sa première base de données

Vous pouvez vérifier si la base de données a été créée avec succès en utilisant la commande :

```
SHOW DATABASES;
```

Pour commencer à travailler avec la base de données nouvellement créée :

```
USE mydatabase;
```

# MySQL : créer sa première table

Une fois connecté à une base de donnée vous pouvez créer votre première table :

```
CREATE TABLE student (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50),
    prénom VARCHAR(50),
    date_naissance DATE,
    adresse VARCHAR(100),
    email VARCHAR(100)
);
```

# Types de données

# MySQL : types de données numériques

- INT : Entier signé.
- FLOAT : Nombre à virgule flottante simple précision.
- DOUBLE : Nombre à virgule flottante double précision.
- DECIMAL : Nombre décimal à précision fixe.

# MySQL : types de données texte

- CHAR : Chaîne de caractères de longueur fixe.
- VARCHAR : Chaîne de caractères de longueur variable.
- TEXT : Texte de longueur variable.

# MySQL : types de données de date et d’heure

- DATE : Date au format "AAAA-MM-JJ".
- TIME : Heure au format "HH:MM:SS".
- DATETIME : Combinaison de date et d'heure au format "AAAA-MM-JJ HH:MM:SS".
- TIMESTAMP : Horodatage au format "AAAA-MM-JJ HH:MM:SS".

# MySQL : types de données booléennes

- BOOLEAN : Valeur booléenne (TRUE/FALSE) ou (0/1).

# MySQL : types de données de clés

- PRIMARY KEY : Clé primaire, utilisée pour identifier de manière unique une ligne dans une table.
- FOREIGN KEY : Clé étrangère, utilisée pour établir des relations entre les tables.

# Constraint

# MySQL : les contraintes

En MySQL, les contraintes sont utilisées pour appliquer des règles et des conditions aux données stockées dans les tables.

# MySQL : les contraintes (clé primaire)

Clé primaire (PRIMARY KEY) :

La contrainte PRIMARY KEY est utilisée pour identifier de manière unique chaque enregistrement dans une table.

Elle garantit que la valeur d'une colonne (ou d'un groupe de colonnes) est unique et non nulle. Une table ne peut avoir qu'une seule clé primaire.

# MySQL : les contraintes (clé secondaire)

Clé étrangère (FOREIGN KEY) :

La contrainte FOREIGN KEY est utilisée pour établir une relation entre deux tables.

Elle garantit que les valeurs d'une colonne (ou d'un groupe de colonnes) d'une table correspondent aux valeurs d'une colonne de référence dans une autre table.

# MySQL : les contraintes (unique)

Contrainte UNIQUE :

La contrainte UNIQUE garantit que les valeurs d'une colonne (ou d'un groupe de colonnes) dans une table sont uniques, à l'exception de la valeur NULL.

Cela signifie qu'aucune autre ligne de la table ne peut avoir la même valeur pour la colonne spécifiée(s).

# MySQL : les contraintes (not null)

Contrainte NOT NULL :

La contrainte NOT NULL garantit qu'une colonne ne peut pas contenir de valeurs nulles.

Cela signifie que chaque enregistrement doit avoir une valeur non nulle pour la colonne spécifiée.

# MySQL : les contraintes (default)

Contrainte DEFAULT :

La contrainte DEFAULT est utilisée pour attribuer une valeur par défaut à une colonne si aucune valeur n'est spécifiée lors de l'insertion d'un nouvel enregistrement.

# MySQL : les contraintes (default)

Lors de la création d'une table, vous pouvez spécifier ces contraintes en utilisant la syntaxe appropriée. Par exemple, pour les contraintes PRIMARY KEY, FOREIGN KEY, NOT NULL et UNIQUE :

```
CREATE TABLE nom_de_la_table (
  id INT PRIMARY KEY,
  email VARCHAR(50) UNIQUE NOT NULL,
  colonne_ref INT,
  FOREIGN KEY (colonne_ref) REFERENCES
autre_table(colonne_reference));
```

# CRUD

# MySQL : insérer des données

Pour insérer des données dans une table MySQL, vous pouvez utiliser la commande INSERT INTO :

```
INSERT INTO nom_de_la_table (colonne1, colonne2, ...)
VALUES (valeur1, valeur2, ...);
```

# MySQL : sélectionner des données

Pour sélectionner des données d'une table MySQL, vous pouvez utiliser la commande SELECT :

```
SELECT * FROM nom_de_la_table;
```

```
SELECT colonne1, colonne2, ... FROM nom_de_la_table;
```

```
SELECT colonne1, colonne2, ... FROM nom_de_la_table
WHERE condition;
```

# MySQL : mettre à jour des données

Pour mettre à jour des données dans une table MySQL, vous pouvez utiliser la commande UPDATE :

```
UPDATE nom_de_la_table
SET colonne1 = nouvelle_valeur1, colonne2 =
nouvelle_valeur2, ...
WHERE condition;
```

# MySQL : mettre à jour des données

Voici un exemple concret :

```
UPDATE employes
SET salaire = 5000, departement = 'RH'
WHERE id = 1;
```

# MySQL : supprimer des données

Pour supprimer des données d'une table MySQL, vous pouvez utiliser la commande DELETE :

```
DELETE FROM nom_de_la_table
WHERE condition;
```

Assurez-vous d'utiliser la clause WHERE avec soin pour spécifier les conditions de suppression de manière précise.
Sinon, vous risquez de supprimer des données indésirables ou de supprimer toutes les lignes de la table par erreur.

# MySQL : alias sur des champs

Pour utiliser un alias sur une colonne et ainsi renommer temporairement le nom de cette dernière dans le résultat d'une requête, il est possible d'utiliser deux syntaxes.

On peut utiliser la commande AS ou non

```
SELECT firstname AS prenom FROM produit;
```

```
SELECT firstname prenom FROM produit;
```

# MySQL : supprimer les doublons

Pour supprimer les doublons des résultats d'une requête en SQL, il est nécessaire d'utiliser la commande SELECT DISTINCT :

```
SELECT DISTINCT prenom FROM etudiant;
```

# MySQL : opérateurs de comparaisons

Il existe de nombreux opérateurs de comparaisons utilisables au sein de la commande WHERE. Le tableau ci-dessous liste les opérateurs les plus couramment utilisés et que nous aborderons dans ce cours :

- =
- != ou <>
- > ou >=
- < ou <=
- IS NULL ou IS NOT NULL

# MySQL : AND et OR

Les opérateurs logiques AND et OR peuvent être utilisés en complément de la commande WHERE pour combiner des conditions :

```
SELECT une_colonne FROM une_table WHERE une_condition AND une_autre_condition;
```

```
SELECT une_colonne FROM une_table WHERE une_condition OR une_autre_condition;
```

# MySQL : BETWEEN

En SQL, l'opérateur BETWEEN permet de sélectionner des enregistrements en fonction d'une intervalle. Cet opérateur s'utilise avec la commande WHERE. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates :

```
SELECT une_colonne FROM une_table WHERE une_colonne
BETWEEN "valeur_1" AND "valeur_2";
```

# MySQL : IN et NOT IN

En SQL, l'opérateur IN permet de sélectionner des enregistrements si la valeur d'une colonne est comprise dans une liste de valeurs. La commande IN permet d'éviter d'avoir à recourir à de multiples OR. Cet opérateur s'utilise avec la commande WHERE :

```
SELECT une_colonne FROM une_table WHERE une_colonne IN
("valeur_1", "valeur_2", "valeur_3");
```

# MySQL : LIKE

En SQL, l'opérateur LIKE permet de faire une recherche suivant un modèle sur les valeurs d'une colonne. Une nouvelle fois, cet opérateur s'utilise avec la commande WHERE. C'est un opérateur extrêmement puissant qui offre de nombreuses possibilités :

```
SELECT * FROM clients WHERE email LIKE "%@yeah.com";
```

L'opérateur LIKE est inséparable des caractères jokers (wildcards) : % (pourcentage) et _ (underscore).
Un caractère joker est utilisé pour se substituer à n'importe quel autre caractère, au sein d'une chaîne de caractères.

# MySQL : alias sur une table

Cette méthode permet d'attribuer un autre nom à une table dans une requête SQL. Cela peut aider à avoir des noms plus courts, plus simples et plus facilement compréhensibles.

```
SELECT * FROM produit AS p;
```

```
SELECT * FROM produit p;
```

# MySQL :  critères de tri

Utilisez la clause ORDER BY pour trier les résultats en fonction d'une colonne spécifiée. Vous pouvez spécifier l'ordre de tri en utilisant "ASC" (par défaut pour un tri croissant) ou "DESC" (pour un tri décroissant).

```
SELECT nom, prenom, salaire
FROM employes
WHERE departement = 'RH'
ORDER BY salaire DESC;
```

# MySQL : limit

La clause "LIMIT" est une instruction utilisée dans MySQL (un système de gestion de base de données relationnelle) pour limiter le nombre de résultats renvoyés par une requête.

Elle permet de spécifier un nombre maximum de lignes à récupérer à partir du début des résultats

```
SELECT * FROM employes
LIMIT 5;
```

# MySQL : limit offset

La clause "LIMIT" peut également être utilisée avec un second argument, spécifiant l'index de départ à partir duquel récupérer les résultats. La syntaxe est la suivante :

```
SELECT * FROM employes
LIMIT 5, 2;
```

# MySQL : fonction d'agrégation

En SQL, les fonctions d’agrégation permettent de réaliser des opérations arithmétiques et statistiques au sein d'une requête. Les principales fonctions sont les suivantes :

- COUNT() compter le nombre d’enregistrements d'une table ou d'une colonne distincte
- AVG() calculer la moyenne sur un ensemble d’enregistrements
- SUM() calculer la somme sur un ensemble d’enregistrements
- MAX() récupérer la valeur maximum d’une colonne sur un ensemble d’enregistrements
- MIN() récupérer la valeur minimum

# Jointure

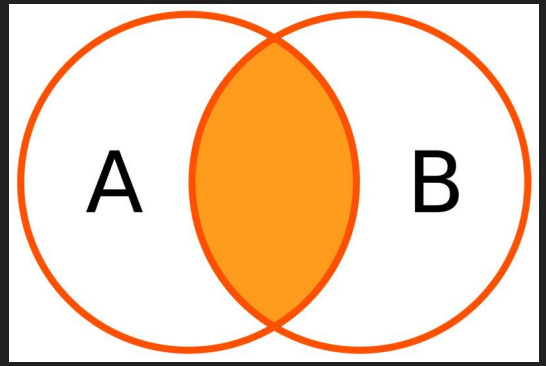
# MySQL : jointure sql

Les jointures en SQL permettent d’associer plusieurs tables dans une même requête.

Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables en même temps de manière efficace.

# MySQL : inner join

Dans le langage SQL la commande INNER JOIN, est un type de jointure très commune pour lier plusieurs tables entre-elles dans une même requête. Cette commande retourne les enregistrements lorsqu'il y a au moins une ligne dans chaque colonne listé dans la condition



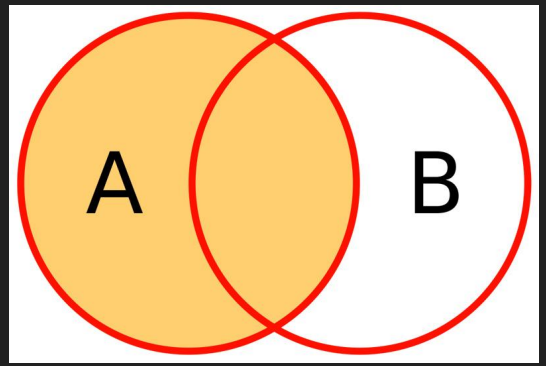
# MySQL : inner join

Cette commande retourne les enregistrements lorsqu'il y a au moins une ligne dans chaque colonne listé dans la condition.

```
SELECT *
FROM table_1
INNER JOIN table_2
ON table_1.une_colonne = table_2.autre_colonne;
```

# MySQL : left join

Cette commande retourne tous les enregistrements de la table première table, celle de gauche (left), avec la correspondance dans la deuxième table si la condition est respectée.



# MySQL : left join

En d'autres termes, ce type de jointure permet de retourner tous les enregistrements d'une table avec les données liées d'une autre table si elles existent.

```
SELECT *
FROM table_1
LEFT JOIN table_2
ON table_1.primary_key = table_2.foreign_key;
```

# Droits Utilisateurs

# MySQL : user root

L'utilisateur "root" est un utilisateur spécial dans MySQL qui possède tous les privilèges et qui a un contrôle total sur le serveur de base de données.
C'est l'utilisateur administrateur par défaut qui est généralement utilisé pour effectuer des tâches d'administration, telles que la création d'autres utilisateurs, l'accès à toutes les bases de données, la modification des paramètres du serveur, etc.

Soyez prudent lorsque vous utilisez cet utilisateur, car il a un accès complet au serveur et peut potentiellement causer des dommages s'il est utilisé de manière incorrecte.

# MySQL : créer un utilisateur

Utilisez la commande CREATE USER pour créer un nouvel utilisateur :

```
CREATE USER 'nom_utilisateur'@'hôte' IDENTIFIED BY
'mot_de_passe';
```

Utilisez '%' pour autoriser toutes les connexions :

```
CREATE USER 'john'@'%' IDENTIFIED BY 'mypass';
```

# MySQL : droit des utilisateurs

Dans MySQL, les utilisateurs sont associés à des droits spécifiques qui déterminent ce qu'ils peuvent faire dans la base de données.

Les droits sont attribués au niveau global, au niveau de la base de données ou au niveau de la table.

# MySQL : accorder tous les privilèges

Pour accorder des droits aux utilisateurs, vous pouvez utiliser la commande GRANT dans MySQL :

```
GRANT ALL PRIVILEGES ON *.* TO 'john'@'%';
```

Par exemple, cette commande accorde tous les privilèges à l'utilisateur "john" sur toutes les bases de données.

# MySQL : accorder certains privilèges

Pour accorder certains droits aux utilisateurs, vous pouvez spécifier les privilèges en précisant avec la commande GRANT dans MySQL :

```
GRANT SELECT ON ma_base_de_donnees.* TO
'utilisateur'@'localhost';
```

Par exemple, cette commande accorde le droit 'SELECT' à un utilisateur sur une base de données connecté en local.

# MySQL : privilèges des utilisateurs

- ALL PRIVILEGES : permet de sélectionner (lire) des données.
- SELECT : permet de sélectionner (lire) des données.
- INSERT : permet d'insérer des données.
- UPDATE : permet de mettre à jour des données.
- DELETE : permet de supprimer des données.
- CREATE : permet de créer de nouvelles bases de données ou tables.
- DROP : permet de supprimer des bases de données ou tables existantes.
- REVOKE : permet de révoquer des droits aux autres utilisateurs.

# MySQL : Lectures complémentaires

Consultez certains des liens pour mieux comprendre le fonctionnement de MySQL:

SQL.sh : Site officiel

Mariadb : Site officiel

Bases de données relationnelles avec SQL : OpenClassRooms

Apprendre SQL de A à Z : Tuto vidéo de Grafikart

# CREATE DATABASE merci;