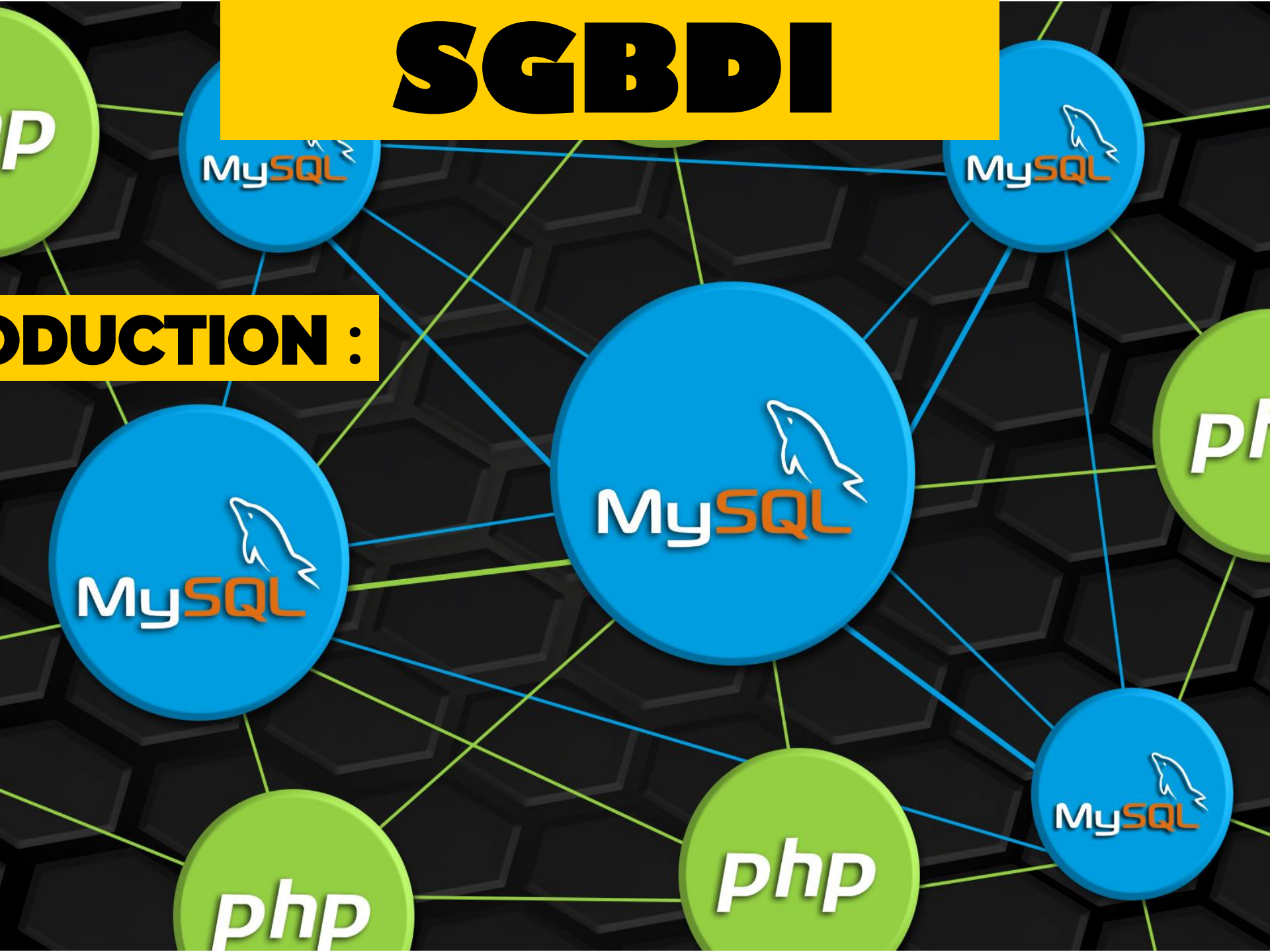
SAGIM

INTRODUCTION AU SGBD

Chapitre 1: Introduction Générale

SGBDI

SGBDI



les SGBD

Un Système de Gestion de Base de Données (**SGBD**) est un logiciel qui permet de **stocker**, **organiser**, **manipuler** et **récupérer** des données de manière **structurée et efficace**. **MySQL** est l'un des SGBD les plus populaires au monde, notamment pour les applications web.

Qu'est-ce que MySQL ?

- **MySQL** est un SGBD relationnel (SGBDR) open-source basé sur le langage SQL (Structured Query Language).
- Il est développé par Oracle Corporation et est largement utilisé pour **les applications web**, les systèmes de gestion de contenu (CMS) comme **WordPress**, et les bases de données en ligne.
- Il est compatible avec de nombreux systèmes d'exploitation (**Windows**, **Linux**, **macOS**) et langages de programmation (**PHP**, Python, Java, etc.).

Fonctionnalités principales de MySQL

- **Gestion des bases de données relationnelles :** Stocke les données dans des tables liées entre elles par des clés primaires et étrangères.
- **Langage SQL** : Utilise des commandes SQL pour créer, lire, mettre à jour et supprimer des données (CRUD : Create, Read, Update, Delete).
- **Transactions** : Supporte les transactions ACID (Atomicité, Cohérence, Isolation, Durabilité) pour garantir l'intégrité des données.
- **Sécurité** : Offre des mécanismes de sécurité avancés (authentification, autorisations, chiffrement).
- **Haute performance** : Optimisé pour les requêtes rapides et les grandes bases de données.

MySQL Database

Créer une base de données :

```
CREATE DATABASE ma_base;
```

Supprimer une base de donneez

```
DROP DATABASE ma_nase;
```

utiliser une base de donneez

```
Use ma_nase;
```

Afficher les bases de donneez:

```
show databases;
```

supprimer une bases de donneez:

```
DROP DATABASE ma_base;
```

Création d'une BD

```
CREATE DATABASE USINE;

USE USINE ;
```

Création d'une table

```
CREATE TABLE departements (
id INT AUTO_INCREMENT PRIMARY KEY,
nom VARCHAR(100) NOT NULL
);
```

Création d'une table

```
CREATE TABLE employes (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nom VARCHAR(50) NOT NULL,
        poste VARCHAR(50),
        salaire DECIMAL(10, 2),
        date_embauche DATE,
        departement_id INT,
        FOREIGN KEY (departement_id)
        REFERENCES departements(id)

);
```

Modification d'une table

```
ALTER TABLE employes
ADD email VARCHAR(100);
```

Modifier une colonne existante :

```
ALTER TABLE employes
MODIFY salaire DECIMAL(12, 2);
```

Supprimer une colonne :

```
ALTER TABLE employes
DROP COLUMN email;
```

Supprimer une Table :

```
DROP TABLE employes;
```

INSERT : Ajouter des données:

```
INSERT INTO departements (nom)
VALUES   ('Informatique'),
         ('Ressources Humaines'),
         ('Marketing');
```

INSERT : Ajouter des données:

```
INSERT
INTO employes (nom, poste, salaire, date_embauche, departement_id)
VALUES
    ('Alice Dupont', 'Développeur', 3500.00, '2022-05-01', 1),
    ('Bob Martin', 'RH Manager', 4500.00, '2021-03-15', 2),
    ('Claire Leroy', 'Resp Marketing', 5000, '2020-09-01', 3);
```

SELECT : Lire les données:

Sélection simple :

```
SELECT * FROM employes;
```

Avec condition :

```
SELECT nom, poste
FROM  employes
WHERE  salaire > 3000;
```

SELECT : Lire les données:

Nombre total d'employés:

```
SELECT COUNT(*) AS total_employes FROM employes;
```

Salaire moyen:

```
SELECT AVG(salaire) AS salaire_moyen FROM employes;
```

Salaire total par poste:

```
SELECT poste, SUM(salaire) AS total_salaire
FROM employes GROUP BY poste;
```

Employé avec le plus grand salaire:

```
SELECT nom, MAX(salaire) AS plus_grand_salaire
FROM employes;
```

UPDATE : Modifier des données

```
UPDATE employes
SET salaire = salaire * 1.10
WHERE poste = 'Développeur';
```

DELETE : Supprimer des données

```
DELETE
FROM employes
WHERE
    date_embauche < '2020-01-01';
```

Combiner SELECT avec fonctions d'agrégat et conditions

```
SELECT poste, COUNT(*) AS nombre_employes,
       AVG(salaire) AS salaire_moyen
FROM employes
GROUP BY poste
HAVING salaire_moyen > 3000;
```

Jointure simple :

Une jointure simple permet de récupérer les employés et leurs départements.

Elle utilise INNER JOIN, qui retourne uniquement les enregistrements où il existe une correspondance entre les deux tables.

```
SELECT      e.nom AS employe, e.poste,
            e.salaire, d.nom AS departement
FROM        employes e
INNER JOIN  departements d
ON          e.departement_id = d.id;
```

Joindre les tables pour afficher les informations des employés et de leurs départements :

```
SELECT e.nom AS employe, e.poste,
e.salaire, d.nom AS departement
FROM employes e
LEFT JOIN
departements d
ON e.departement_id = d.id;
```

SAGIM

SGBD

TP de synthèse:
Gestion d’une bibliothèque.

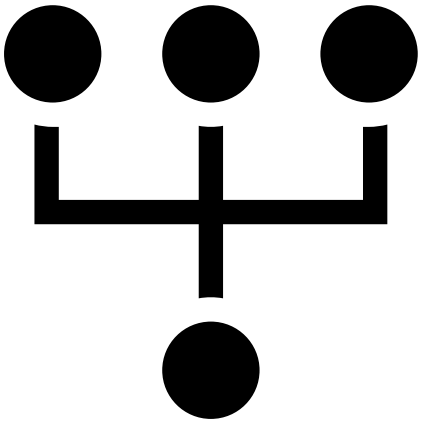
Gestion d'une bibliothèque:

Créer une base de données pour gérer une bibliothèque.

La base de données doit contenir des informations sur les livres, les auteurs, et les emprunts de livres par les membres de la bibliothèque.

Création de la base de données:

```
CREATE DATABASE bibliotheque;
USE bibliotheque;
```



Création des tables:

Table Auteurs : Contient les informations sur les auteurs.

Table Livres : Contient les informations sur les livres.

Table Membres : Contient les informations sur les membres de la bibliothèque.

Table Emprunts : Contient les informations sur les emprunts de livres par les membres.

Création des tables:

```
CREATE TABLE Auteurs (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    nationalite VARCHAR(50)
);
```

Création des tables:

```
CREATE TABLE Livres (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titre VARCHAR(200) NOT NULL,
    id_auteur INT,
    annee_publication INT,
    FOREIGN KEY (id_auteur) REFERENCES Auteurs(id)
);
```

Création des tables:

```
CREATE TABLE Membres (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    date_inscription DATE
);
```

Création des tables:

```
CREATE TABLE Emprunts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    id_livre INT,
    id_membre INT,
    date_emprunt DATE,
    date_retour DATE,
    FOREIGN KEY (id_livre) REFERENCES Livres(id),
    FOREIGN KEY (id_membre) REFERENCES Membres(id)
);
```

Insertion des auteurs :

```
INSERT INTO Auteurs (nom, nationalite)
VALUES
        ('George Orwell', 'Britannique'),
        ('J.K. Rowling', 'Britannique'),
        ('Haruki Murakami', 'Japonais');
```

Insertion des livres:

```
INSERT INTO Livres (titre, id_auteur, annee_publication)
VALUES
        ('1984', 1, 1949),
        ('Animal Farm', 1, 1945),
        ('Harry Potter à l\'école des sorciers', 2, 1997),
        ('Kafka sur le rivage', 3, 2002);
```

Insertion des membres:

```
INSERT INTO Membres (nom, email, date_inscription)
VALUES
('Alice Dupont', 'alice.dupont@example.com', '2023-01-15'),
('Bob Martin', 'bob.martin@example.com', '2023-02-20');
```

Insertion des emprunts:

```
INSERT INTO Emprunts (id_livre, id_membre,
                      date_emprunt, date_retour)
VALUES
        (1, 1, '2023-03-01', '2023-03-15'),
        (3, 2, '2023-03-05', NULL);
```

SAGIM

Introduction
Aux SGBD I

Requettes d’interrogation simples et avec jointure …

SGBDI

Mysql server

Requette 1:

Compter le nombre total de livres dans la bibliothèque :

```
SELECT COUNT(*) AS total_livres
FROM Livres;
```

Requette 2:

Trouver tous les membres qui se sont inscrits en 2023 :

```
SELECT COUNT(*) AS total_livres
FROM Livres;
```

Requette 3:

Sélectionner tous les livres publiés après l'année 2000 :

```
SELECT titre, annee_publication
FROM Livres
WHERE annee_publication > 2000;
```

Requette 4:

Liste de tous les livres avec leurs auteurs :

```
SELECT Livres.titre, Auteurs.nom AS auteur
FROM Livres
JOIN Auteurs ON Livres.id_auteur = Auteurs.id;
```

Requette 5:

Lister tous les auteurs et le nombre de livres qu'ils ont écrits :

```
SELECT
    Auteurs.nom, COUNT(Livres.id) AS nombre_de_livres
FROM
    Auteurs
LEFT JOIN
    Livres ON Auteurs.id = Livres.id_auteur
GROUP BY
    Auteurs.id;
```

Requette 6:

Trouver le livre le plus ancien dans la bibliothèque :

```
SELECT titre, annee_publication
FROM Livres
ORDER BY annee_publication ASC
LIMIT 1;
```

Requette 7:

Trouver tous les livres qui n'ont jamais été empruntés :

```
SELECT Livres.titre
FROM Livres
LEFT JOIN Emprunts
ON Livres.id = Emprunts.id_livre
WHERE Emprunts.id_livre IS NULL;
```

Requette 8:

Mettre à jour l'année de publication d'un livre spécifique (par exemple, changer l'année de publication de "1984" à 1948) :

```
UPDATE Livres
SET annee_publication = 1948
WHERE titre = '1984';
```

Requette 9:

Supprimer tous les emprunts qui ont été retournés avant une certaine date (par exemple, avant le 2023-03-10) :

```
DELETE FROM Emprunts
WHERE date_retour < '2023-03-10';
```