

Les bases de SQL

La **bases de données** (BDD) sont gérées par des SGBD (Systèmes de **G**estion de **B**ases de **D**onnées) qui sont des programmes qui se chargent du stockage des données. Nous utiliserons le SGBD **MySQL**.
Pour communiquer avec MySQL, on utilise le langage SQL.

CRUD :

- Create (**INSERT INTO** en sql)
- read (**SELECT** en sql)
- update (**UPDATE** en sql)
- Delete (**DELETE** en sql)

I) Lire la BDD avec SELECT

SELECT * FROM utilisateur

- **SELECT** : demande à MySQL d'afficher ce que contient une table.
- ***** : indique quels champs MySQL doit récupérer dans la table (***** pour récupérer tous les champs).
- **FROM utilisateur** : indique à MySQL dans quelle table (ici **utilisateur**) on va récupérer les données.

SELECT nom, prenom FROM utilisateur

- **nom, prenom** : pour récupérer uniquement les champs **nom** et le **prenom**.

Critères de sélection : clause WHERE

SELECT * FROM utilisateur WHERE id=5;

On peut combiner plusieurs conditions avec **AND** et **OR** : **SELECT * FROM utilisateur WHERE age>25 AND sex="M"**

Ordonné : clause ORDER BY

SELECT * FROM utilisateur ORDER BY nom ASC (ou **DESC**)

On peut combiner avec la clause **WHERE** : **SELECT * FROM utilisateur WHERE sex="F" ORDER BY nom ASC**

Limité : clause LIMIT

SELECT * FROM utilisateur LIMIT 10 : affiche les 10 premiers

SELECT * FROM utilisateur LIMIT 5, 10 : affiche 10 lignes à partir du 6ème

On peut combiner avec la clause **WHERE** et **ORDER BY** :

SELECT * FROM utilisateur WHERE sex="F" ORDER BY nom ASC LIMIT 2

Attention : Il est impératif de placer les mots-clés dans l'ordre : **WHERE**, puis **ORDER BY** et enfin **LIMIT**.

II) Modifier la BDD avec INSERT INTO, UPDATE ET DELETE

Ajouter des données : INSERT INTO

INSERT INTO utilisateur (nom, prenom, age, sex) VALUES ('Rabbit', 'Roger', 16, 'M')

Attention : On ne saisit pas le champ **id**, car il a la propriété **AUTO_INCREMENT**.

Modifier des données : UPDATE

UPDATE utilisateur SET prenom= 'jeanne', sex='F' WHERE id=5

Attention : si on oublie la clause **WHERE**, toutes les entrées seront modifiées.

Supprimer des données : DELETE

DELETE FROM utilisateur WHERE nom='Dupont'

Attention : si on oublie le **WHERE**, la table sera entièrement vidée.

III) Communiquer avec la BDD en PHP

Lire des données dans la BDD en PHP

Pour communiquer avec la BDD en PHP on va utiliser un objet PDO

```
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
```

On injecte la requête SQL dans l'objet PDO

```
$request = $pdo->query('SELECT * FROM utilisateur');
```

On récupère les données

```
$users = $request->fetchAll(PDO::FETCH_ASSOC); // FETCH_ASSOC pour récupérer un tableau associatif
```

On peut afficher les utilisateurs un par un avec un foreach

```
foreach ($users as $user) {  
    echo $user['nom'] . ' ' . $user['prenom'] . '<br>';  
}
```

Remarque : pour récupérer plusieurs lignes, on utilise `fetchAll()` et pour récupérer une seule ligne on utilise `fetch()`.

Attention : quand on injecte des variables dans une requête, on utilise des **requêtes préparées** pour se protéger de **l'injection SQL**.

```
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
```

```
$request = $pdo->prepare('SELECT * FROM utilisateur WHERE nom=:nom'); // paramètre nommé (:nom)
```

```
$request->execute(['nom' => 'Jean']); // on donne une valeur au paramètre nommé (:nom)
```

```
$user = $request->fetch(PDO::FETCH_ASSOC); // ici on utilise fetch() car on ne récupère qu'une seule ligne
```

Modifier la BDD en PHP

Pour modifier la BDD on utilise des requêtes préparées.

```
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'user', 'password');
```

```
$sql = '.....' // requête INSERT INTO, UPDATE ou DELETE avec paramètres nommés
```

```
$request = $pdo->prepare($sql);
```

```
$request->execute(['param1' => $value1, 'param2' => $value2]);
```

IV) Les fonctions SQL

Les fonctions scalaires

`SELECT UPPER(nom) FROM utilisateur` : renvoie le nom en majuscule

Autres exemples : `LOWER()`, `LENGTH()`, `CEIL()`, ...

On peut utiliser un alias : `SELECT UPPER(nom) as nom_maj FROM utilisateur`

Les fonctions d'agrégations : Elles ne renvoient qu'une seule valeur

`SELECT COUNT(*) FROM utilisateur` : renvoie le nombre de ligne

avec alias : `SELECT COUNT(*) as nombre_utilisateur FROM utilisateur`

Autres exemples : `MAX()`, `MIN()`, `AVG()`, ...