

Les bases de MySQL

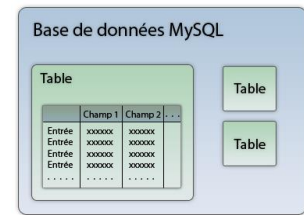
I) Les bases de données

1) Qu'est-ce qu'une base de données

Une **base de données** est l'endroit où sont stockées les données de manière organisée. Elle est composée d'une ou plusieurs **tables**.

Chaque **table** contient des :

- **Champs** : colonnes
- **Entrées** : lignes



2) PhpMyAdmin

Création d'une base de données

- Cliquer sur "Nouvelle base de données".
- Entrer le nom que vous souhaitez donner à votre base de données
- Cliquer sur "créer".

Création d'une nouvelle table

- **Champ** : permet de définir le nom du champ
- **Type** : le type de données que va stocker le champ (**INT**, **VARCHAR**, **TEXT**, **DATE**, etc.)
- **Taille/Valeurs** : permet d'indiquer la taille maximale du champ (utile pour le type **VARCHAR**)
- **Index** : active l'indexation du champ.
- **AUTO_INCREMENT** ou **A_I** : pour incrémenter automatiquement le champ à chaque nouvelle entrée

Les clés primaires : permettent une identification unique de chaque entrée.

Exemple : **id** Numéro d'identification unique qui permettra de numérotter toutes les entrées (type **INT**, index **PRIMARY** et **AUTO_INCREMENT** Coché).

Autres opérations

Onglet « SQL » : Pour taper des requêtes SQL en langage SQL.

Onglet « Importer » : Pour importer une BDD ou des requêtes SQL.

Onglet « Exporter » : Pour Transférer ou sauvegarder la BDD.

Onglet « Opérations » : On peut effectuer diverses opérations sur la table comme : Renommer la table en / Déplacer la table vers / Copier la table vers / Optimiser la table / Vider / Supprimer.

II) Le langage SQL

Le sql est divisé en deux familles :

- Le langage de définition de données (LDD) pour créer et définir la structure de la BDD (équivalent des opérations faites avec l'interface graphique phpMyAdmin).
- Le langage de manipulation de données (LMD) : CRUD
 - C : create pour ajouter des données à la BDD (INSERT INTO).
 - R : read pour sélectionner des données dans la BDD (SELECT).
 - U : update pour modifier des données dans la BDD (UPDATE).
 - D : delete pour supprimer des données dans la BDD (DELETE).

1) Les requêtes SQL du CRUD

SELECT : pour récupérer des données

SELECT champ1, champ2, ... FROM nom_table

INSERT : pour ajouter des données

INSERT INTO nom_table (champ1, champ2, ...) VALUES ('valeur1','valeur2',...)

UPDATE : pour modifier des données

UPDATE nom_table SET champ1 = 'valeur1', champ2='valeur2' WHERE //filtrage désiré

Attention : si on oublie **WHERE**, toutes les entrées seront affectées.

DELETE : supprimer des données (irréversible !)

DELETE FROM nom_table WHERE //filtrage désiré

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

2) Les critères de sélection

Filtré : permet de trier les données

SELECT * FROM nom_table WHERE entree='valeur_entree'

Ordonné : permet d'ordonner les résultats

ORDER BY : **SELECT * FROM nom_table ORDER BY entree**

Pour classer par ordre décroissant, écrire : **SELECT * FROM nom_table ORDER BY entree DESC**

Limité : permet de ne sélectionner qu'une partie des résultats.

SELECT * FROM nom_table LIMIT n, m

n : n^{ème} entrée (0= 1^{ère} entrée) et m : nombre d'entrée à sélectionner.

Plusieurs requêtes en une seule :

SELECT champs1, champs2 FROM table WHERE champs1='val1' OR champs2='val2' ORDER BY champ3 DESC LIMIT 0,10

On doit placer les mots-clés dans l'ordre : **WHERE**, puis **ORDER BY** et enfin **LIMIT**.

III) Communiquer avec la base de données en PHP

1) Connexion à la base de données avec PDO

Pour se connecter à une BDD MySQL, on utilisera l'extension **PDO** (extension orientée objet utilisable sur toutes les BDD).

Connexion à MySQL avec PDO : **\$bdd = new PDO('mysql:host=localhost; dbname=test; charset=utf8', 'root', 'mon_pass');**

- host : c'est l'adresse de l'ordinateur où MySQL est installé (ici **localhost** car MySQL et PHP sont installés sur le même ordinateur).
- dbname : c'est le nom de la base de données à laquelle vous voulez vous connecter (ici **test**).
- Le login : Le plus souvent, c'est le même login que celui utilisé pour le FTP (ici **'root'**)
- Le mot de passe : le plus souvent, le mot de passe est le même que celui utilisé pour accéder au FTP (ici **'mon_pass'**).

2) Faire une requête avec PDO

Ecrire une requête avec PDO : **\$reponse = \$bdd->query('Requête SQL ici');** //on récupère la réponse dans l'objet **\$reponse**.

Afficher le résultat d'une requête :

```
<?php
while ($donnees = $reponse->fetch())           //fetch() renvoie la première entrée
{                                                // $donnees==false après avoir passé la dernière entrée
//Utilisation de $donnees
}
$reponse->closeCursor(); // provoque la « fermeture du curseur d'analyse des résultats ».
?>
```

3) Construire des requêtes en fonction de variables

Il ne faut pas concaténer une variable dans une requête (risque d'injection SQL) :

~~**\$reponse = \$bdd->query('SELECT champ FROM table WHERE champ="' . \$_GET[\$data] . '");**~~

Pour se protéger de l'injection SQL, on utilise les requêtes préparées plus sûres et plus rapides.

On prépare la requête avec **prepare()** puis on l'exécute avec **execute()**

\$req = \$bdd->prepare('SELECT * FROM table WHERE champ1= :valeur1 AND champ2= :value2');

```
$req->execute([
    'champs1' => valeur1,
    'champs2' => value2,
]);
```

4) Equivalence entre entité et table (en POO)

Dans la base de données :

- **user** est une **table** (sans majuscule)
- **username** et **password** sont des **champs** (colonnes)

user
id: INTEGER NOT NULL [PK]
username: VARCHAR(255) NOT NULL [AK]
password: VARCHAR(255) NOT NULL

Dans notre script PHP :

- **User** est une **entité**
- **Username** et **password** sont des **attributs**

User
- id : int
- username : string
- password : string

Dans la table **user**, chaque **ligne** représente une **instance** de l'entité **User**.

Aller plus loin avec le langage SQL

1) Les fonctions SQL (notation en majuscules)

Les fonctions scalaires : Elles agissent sur chaque entrée.

Exemple :

<code>SELECT UPPER(nom) AS nom_maj FROM jeux_video</code>	: « O'Clock » → « O'CLOCK »
<code>SELECT LOWER(nom) AS nom_min FROM jeux_video</code>	: « O'Clock » → « o'clock »
<code>SELECT LENGTH(nom) AS longueur_nom FROM jeux_video</code>	: « O'Clock » → 7
<code>SELECT ROUND(prix, 2) AS prix_arrondi FROM jeux_video</code>	: 12,3221 → 12,32

Les fonctions d'agrégat : Elles retournent une valeur UNIQUE après avoir fait des calculs sur l'ensemble de la BDD.

Exemple :

<code>SELECT AVG(note) AS moyenne FROM carnet_de_note</code>	→ Calcule la moyenne
<code>SELECT COUNT(*) AS nbre_note FROM carnet_note</code>	→ Compte le nombre d'entrées d'un champ
<code>SELECT SUM(prix) AS prix_total FROM articles</code>	→ Additionne les valeurs d'un champ
<code>SELECT MAX(prix) AS prix_max FROM articles</code>	→ Retourne la valeur maximale d'un champ
<code>SELECT MIN(prix) AS prix_min FROM articles</code>	→ Retourne la valeur minimale d'un champ

Le groupement de données : GROUP BY et HAVING

Grouper des données : avec **GROUP BY** (utilisé en combinaison d'une fonction d'agrégat)

`SELECT AVG(prix) AS prix_moyen, fruit FROM article GROUP BY fruit` → Prix moyen par fruit

Filtrer des données regroupées : avec **HAVING** (c'est un **WHERE** qui agit sur des données regroupées après un **GROUP BY**).

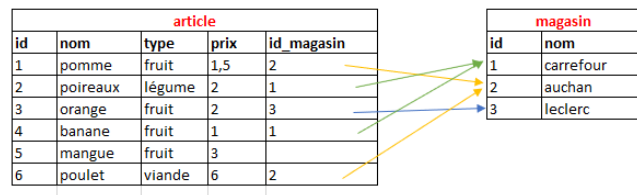
`SELECT AVG(prix) AS prix_moyen, fruit FROM article GROUP BY fruit HAVING prix_moyen <= 1`

→ Moyennes des prix inférieurs à 1€ par type de fruit.

2) Les jointures entre table

Les tables **article** et **magasin** sont reliées à travers **id** et **id_magasin**.

Il est possible de joindre ces deux tables avec une requête jointe SQL.



Les jointures internes : ne sélectionnent que les données qui ont une correspondance entre les deux tables

```
SELECT a.nom nom_article, prix, m.nom nom_magasin
FROM article a
INNER JOIN magasin m
ON a.id_magasin = m.id
```

On peut ajouter les filtres à la fin de la requête

```
WHERE j.magasin = 'auchan'
ORDER BY prix DESC
LIMIT 0, 3
```

nom	prix	nom_magasin
pomme	1,5	auchan
poireaux	2	carrefour
orange	2	leclerc
banane	1	carrefour
poulet	6	auchan

Avec le **JOIN**, on récupère les données depuis une table principale (**article**) et on fait une jointure interne (**INNER JOIN**) avec une autre table (**magasin**). La liaison entre les champs est faite dans la clause **ON** la ligne suivante.

Les jointures externes : sélectionnent toutes les données, même celles sans correspondance.

```
SELECT a.nom nom_article, prix, m.nom nom_magasin
FROM article a
LEFT JOIN magasin m
ON a.id_magasin = m.id
```

Le **LEFT JOIN** demande à récupérer tout le contenu de la table de gauche et le joint au contenu de la table de droite. S'il n'y a pas d'équivalence dans la table de droite, **NULL** y sera stocké.

Avec **RIGHT JOIN**, c'est la table de droite qui sera intégralement récupérée.

nom	prix	nom_magasin
pomme	1,5	auchan
poireaux	2	carrefour
orange	2	leclerc
banane	1	carrefour
mangue	3	NULL
poulet	6	auchan

Même chose avec **RIGHT JOIN**, sauf que c'est **jeux_video** qui est récupéré.