Les bases de données et PHP

Cyril Rabat
cyril.rabat@univ-reims.fr

Licence 2 Informatique - Info0303 - Programmation Web 2

2023-2024

Cours n°4

Présentation des bases de données, un peu de SQL
Utilisation des bases de données en PHP

Version 18 septembre 2023

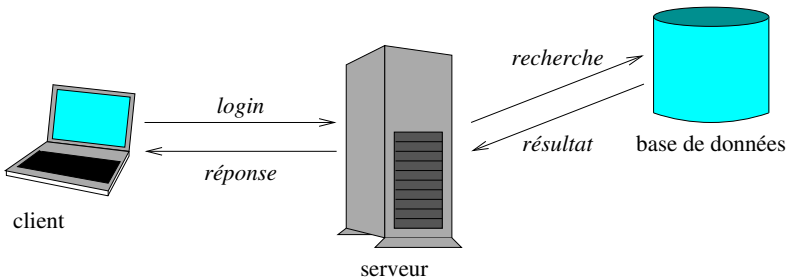
Table des matières

1. Création et manipulation d'une base de données
   - Introduction sur les bases de données
   - Mise en place d'une base de données
   - Manipulation des données avec SQL
2. Manipulation de bases de données en PHP
   - Introduction
   - PDO
   - *mysqli*
3. Conclusion

À quoi sert une base de données ?

- Permet de stocker des données indexées :
  ↪ $\neq$ fichiers à plat
- Plusieurs types de bases de données :
  - Bases de données relationnelles
    ↪ Exemples : MySQL, Oracle, SQLServeur, PostgreSQL, *etc.*
  - Bases de données noSQL :
    ↪ Exemples : MongoDB, BigTable, Cassandra, *etc.*
- SGBD : Système de Gestion de Base de Données
- Interactions avec les SGBD relationnelles :
  - Utilisation du langage SQL (*Structured Query Language*)
  - Permet : création, recherches, combinaisons, tris, *etc.*

Intéractions avec une base de données



Le langage SQL

- SQL pour *Structured Query Language*
- Langage utilisé pour interagir avec les SGBD :
  - Modification de la structure : création, suppression de tables
  - Manipulation des données : ajout, modification, suppression
- Standardisé
  ↪ Mais différentes versions !
  ↪ Syntaxe variable suivant les SGBD

Comment créer une base de données ?

- Connexion au SGBD puis saisie des commandes via le CLI
  ↪ Utilisation du SQL
- Utilisation d'une application
  ↪ Exemple : *phpMyAdmin* (application Web en PHP), *HeidiSQL* (client lourd *Windows*)
- Les outils permettent :
  - L'importation et l'exportation de la base (structures et/ou données)
  - La création et la modification des tables simplifiées
  - L'exécution des requêtes SQL
- Avec une distribution comme *Wamp* ou *EasyPHP* :
  ↪ Accès par défaut : `http://localhost/phpmyadmin/`
  ↪ Par défaut, connexion : login = root ; mot de passe = aucun ou root

Première base de données

- Avec *phpMyAdmin* :
  - Nouvelle base de données
    ↪ Nom : INFO0303
    ↪ Interclassement : `utf8mb4_0900_ai_ci` (par exemple)
- Création de la table :
  - Création de tables en mode graphique :
    - Nom des champs, types, etc. dans des formulaires
    - Moteur de stockage :
      ↪ *MyISAM* ou *InnoDB* (les autres ne seront pas traités)
  - En SQL :
    - Sélection de la base
    - Onglet SQL
    - Saisie de la requête

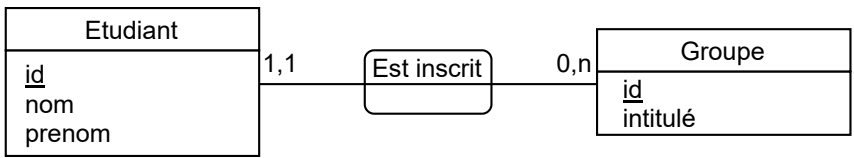
Moteur de stockage : *MyISAM* versus *InnoDB*

- *MyISAM* :
  - Stockage par défaut dans *MySQL*
  - Avantages :
    - Rapide pour SELECT (sélection) et INSERT (insertion)
    - Indexation plein texte ; meilleure performance sur la recherche de texte
    - Plus souple au niveau de l'intégrité des données
  - Inconvénients : pas de clefs étrangères
- *InnoDB* :
  - Avantages :
    - Gestion des clefs étrangères et des contraintes d'intégrité
    - Gestion des transactions
    - Système de récupération en cas de crash
  - Inconvénients :
    - Pas d'indexation plein texte
    - Demande plus de ressources, plus lent

Exemple : les étudiants

- Un étudiant est associé à un groupe
- Table Groupe(id, intitulé)
  - Clé primaire : id
- Table Etudiant (id, nom, prénom, groupe)
  - Clé primaire : id
  - Clé étrangère : groupe qui référence Groupe(id)

Le MCD



- Un étudiant appartient à un groupe et un seul
- Un groupe contient 0 ou plusieurs étudiants

Types des données

- Groupe :
  - id : entier
    ↪ `unsigned int`
  - intitulé : chaîne de caractères
    ↪ `varchar(100)`
- Etudiant :
  - id : entier
    ↪ `unsigned int`
  - nom, prénom : chaînes de caractères
    ↪ `varchar(100)`
  - groupe : entier
    ↪ `unsigned int`

Création d'une table en SQL

```
CREATE TABLE [IF NOT EXISTS] Nom_Table (
  Nom_Colonne Type [NOT NULL | NULL] [DEFAULT Valeur_Defaut] [AUTO_INCREMENT]
  ...
  [CONSTRAINT Nom_Clef_Primaire] PRIMARY KEY (Nom_Colonne_1, ...)
  [CONSTRAINT Nom_Clef_Etrangère] FOREIGN KEY (Nom_Colonne_1,...)
      REFERENCES Nom_Table (Nom_Colonne_1, ...)
)
[DEFAULT] CHARACTER SET = Encodage
ENGINE = InnoDB | MyISAM
```

- Possible de créer les clefs ensuite
  ↪ `ALTER TABLE`
- Types de colonne :
  ↪ INT, TINYINT, SMALLINT, FLOAT, DOUBLE ...
  ↪ VARCHAR(taille), TEXT ...
  ↪ DATE, TIME, ...

Création des tables `etudiant` et `groupe` (1/2)

```
CREATE TABLE `groupe` (
  `id` int UNSIGNED NOT NULL AUTO_INCREMENT,
  `intitule` varchar(100) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `etudiant` (
  `id` int UNSIGNED NOT NULL AUTO_INCREMENT,
  `nom` varchar(100) NOT NULL,
  `prenom` varchar(100) NOT NULL,
   `groupe` int UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  FOREIGN KEY (`groupe`) REFERENCES `groupe` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Création des tables `etudiant` et `groupe` (2/2)

Vision depuis le concepteur de phpmyadmin

Ajout de données

```
INSERT INTO `Nom_Table` (`Champ_1`, `Champ_2`, ..., `Champ_N`)
VALUES (Valeur_1, Valeur_2, ..., Valeur_N);
```

- Ajout d'un enregistrement dans une table
- Plusieurs enregistrements possibles (séparés par ',')

```
INSERT INTO `groupe` (`id`, `intitule`) VALUES
  (NULL, "S3F3"),
  (NULL, "S3F4"),
  (NULL, "S3F5");

INSERT INTO `etudiant` (`id`, `nom`, `prenom`, `groupe`) VALUES
  (NULL, "Schwarzenegger", "Arnold", 1),
  (NULL, "Eastwood", "Clint", 2),
  (NULL, "Stallone", "Sylvester", 1);
```

Suppression de données

```
DELETE FROM `Nom_Table` WHERE conditions
```

- Suppression de tous les enregistrements correspondant aux conditions
- Possible de spécifier une limite
  ↪ `LIMIT 1` : un seul enregistrement sera supprimé
- Condition `WHERE 1` : tous les enregistrements sont supprimés !

```
/* Suppression de l'étudiant d'identifiant 1 */
DELETE FROM `etudiant` WHERE `id` = 1

/* Arnold Schwarzenegger a été effacé... */
/* ...mais il reviendra */
```

Mise-à-jour de données

```
UPDATE `Nom_Table`
SET `Col_1`=Nouv_Val_1[, Col_2=Nouv_Val_2[, ..., Col_N = Nouv_Val_N]]
[WHERE Condition]
```

- Mise-à-jour de tous les enregistrements en fonction de la condition

```
/* Changement de groupe des étudiants du groupe 2 */
UPDATE `etudiant` SET `groupe`=1 WHERE `groupe`=2
```

Sélection de données (1/4)

```
SELECT * FROM `Nom_Table` [WHERE Conditions]
```

- Sélectionne toutes les données de la table
- Possible de spécifier une condition

Exemple

```
SELECT * FROM `etudiant`
```

Résultat :

id	nom	prenom	groupe
1	Schwarzenegger	Arnold	1
2	Eastwood	Clint	2
3	Stallone	Sylvester	1
4	Norris	Chuck	1

Sélection de données (2/4)

```
SELECT `Champ_1`, ..., `Champ_N` FROM `Nom_Table` [WHERE Conditions]
```

- Possible de spécifier des champs/colonnes spécifiques

Exemple

```
SELECT `nom`, `prenom` FROM `etudiant`
```

Résultat :

nom	prenom
Schwarzenegger	Arnold
Eastwood	Clint
Stallone	Sylvester
Norris	Chuck

Sélection de données (3/4)

```
SELECT `Champ_1`, ..., `Champ_N` FROM `Nom_Table_1`, `Nom_Table_2` WHERE conditions
```

- Sélectionne des données de plusieurs tables
  ↪ Jointure

Exemple

```
SELECT `nom`, `prenom`, `intitule` FROM `etudiant`, `groupe`
WHERE `groupe`=`id`
```

Résultat :

nom	prenom	intitule
Schwarzenegger	Arnold	S3F3
Eastwood	Clint	S3F4
Stallone	Sylvester	S3F3
Norris	Chuck	S3F3

Sélection de données (4/4)

```
SELECT `Champ_1` as alias1, ..., `Champ_N` FROM `Nom_Table_1`, `Nom_Table_2` WHERE
    conditions
```

- Utilisation des alias en cas d'ambiguïté

Exemple

```
SELECT A.`nom`, A.`prenom`, B.`intitule` AS `groupe`
FROM `etudiant` AS A, `groupe` AS B WHERE A.`groupe`=B.`id`
```

Résultat :

nom	prenom	groupe
Schwarzenegger	Arnold	S3F3
Eastwood	Clint	S3F4
Stallone	Sylvester	S3F3
Norris	Chuck	S3F3

Remarques

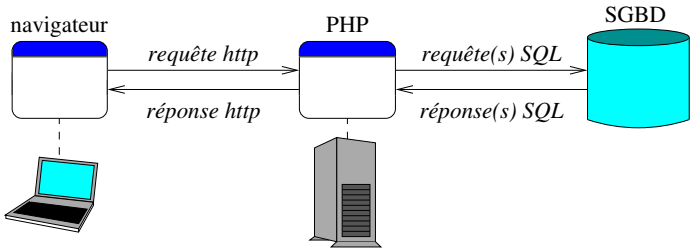
- Les requêtes peuvent devenir très vite complexes !
- Solutions :
  - Réfléchir au préalable à la structure de la base (MCD)
  - Utiliser une notation spécifique
    ↪ Utilisation de préfixes, par exemple

Exemple


```
SELECT `etu_nom`, `etu_prenom`, `grp_intitule`
FROM `etudiant`, `groupe` WHERE `etu_groupe`=`grp_id`
```

Introduction

- La base de données est utilisée pour stocker des données
- Le script PHP :
  - Exploite la base de données (stockage, récupération)
  - Produit du contenu dynamique à partir des données



API disponibles en PHP

- PDO (pour *PHP Data Objects*) :
  - Avantages :
    - Code indépendant du SGBD
    - Simple, portable
  - Inconvénients :
    - Ne permet pas de profiter pleinement des fonctionnalités d'un SGBD
- *mysqli* :
  - API spécifique à MySQL
  - Recommandée pour l'utilisation de MySQL version 4.7 et plus
    ↪ Version fournie par Wamp : 8.0.26 (au 15/09/2023)

Ne pas utiliser les fonctions `mysql_*` qui sont obsolètes.

Connexion à une base de données *MySQL*

```
define("BD_HOST", "localhost");
define("BD_BASE", "info0303");
define("BD_USER", "root");
define("BD_PASSWORD", "");

try {
    $BD = new PDO(
        "mysql:host=".BD_HOST.";dbname=".BD_BASE.";charset=UTF8",
                BD_USER, BD_PASSWORD
      );
} catch(Exception $e) {
    echo "<p> Problème de connexion à la base de données. </p>";
    die();
}
```

- Cela correspond à la configuration par défaut
- `$BD` est l'objet PDO correspondant à la connexion avec la base
- Sans `dbname=".DB_BASE."` : connexion au SGBD
  ↪ Utile pour créer une base de données

Sélection (1/2)

```
$SQL = <<<SQL
SELECT `etu_nom`, `etu_prenom`, `grp_nom`
FROM `etudiant`, `groupe` WHERE `etu_groupe`=`grp_id`;
SQL;

if($requete = $BD->query($SQL)) {
    echo "<ul>";
    while($ligne = $requete->fetch()) {
        echo <<<HTML
  <li>
    <b>{$ligne["etu_nom"]} {$ligne["etu_prenom"]}</b> : {$ligne['grp_nom']}
  </li>
HTML;
    }
    echo "</ul>";
}
else
    echo "<p>␣Erreur␣lors␣de␣l'exécution␣de␣la␣requête.␣</p>";
```

- La méthode `PDO::query` retourne un objet `PDOStatement`
- La méthode `PDOStatement::fetch` retourne un tableau associatif

Sélection (2/2)

- La méthode `fetch` retourne un tableau associatif :
  ↪ Par défaut, données associées au nom de colonne + au numéro
  ↪ `[ 'etu_nom' => "Schwarzenegger", 0 => "Schwarzenegger", ...]`
- Possible de spécifier d'autres modes :
  - `FETCH_BOTH` : par défaut
  - `FETCH_ASSOC` : valeurs indexées par le nom de la colonne
  - `FETCH_NUM` : valeurs indexées par le numéro de la colonne
- Possible aussi de créer des objets directement

Autres requêtes

```
$SQL = <<<SQL
INSERT INTO `etudiant` (`etu_id`, `etu_nom`, `etu_prenom`, `etu_groupe`)
VALUES (NULL, 'De Niro', 'Robert', 2);
SQL;

if($requete = $BD->exec($SQL))
  echo "<p> Données ajoutées. </p>";
else
  echo "<p> Erreur lors de l'ajout. </p>";
```

- `PDO::exec` retourne un nombre de lignes (ou FALSE en cas d'erreur)
- Utilisable pour `INSERT`, `DELETE`, `UPDATE`

Requêtes préparées (1/2)

- Possible de préparer des requêtes (**recommandé**) :
  - Utile lorsque des requêtes sont exécutées plusieurs fois
  - Évite du temps de préparation à la base
  - Limite les problèmes d'injection
- Exécution de la requête en deux temps :
  1. Préparation de la requête
  2. Exécution de la requête à partir de paramètres
     ↪ Plusieurs fois si nécessaire
- Préparation avec la méthode `PDO::prepare`
- Exécution avec `PDOStatement::execute`
  ↪ Possible d'ajouter les valeurs des paramètres

Requêtes préparées (2/2)

```
$SQL = <<<SQL
SELECT `etu_nom`, `etu_prenom`, `grp_intitule`
FROM `etudiant`, `groupe`
WHERE `etu_groupe`=`grp_id` AND `etu_nom`= :nom;
SQL;

if($requete = $BD->prepare($SQL)) {
  if($requete->execute([':nom' => "Schwarzenegger"])) {
    echo "<ul>";
    while($ligne = $requete->fetch(PDO::FETCH_ASSOC))
      echo <<<HTML
<li><b>{$ligne['etu_nom']} {$ligne['etu_prenom']}</b> : {$ligne['grp_intitule']} </li>
HTML;
    echo "</ul>";
  }
  else
    echo "<p>␣Erreur␣lors␣de␣l'exécution␣de␣la␣requête.␣</p>";
}
else
  echo "<p>␣Erreur␣lors␣de␣la␣préparation␣de␣la␣requête.␣</p>";
```

Les paramètres d'une requête préparée

- Possible d'utiliser les '`:nom`' (avec "nom" le nom du paramètre)
  ↪ Paramètres nommés
- Possible d'utiliser les '?' :
  ↪ Remplacés dans l'ordre d'apparition dans la requête
- Pour spécifier les paramètres :
  - Directement lors de l'appel à la méthode `execute`
  - Avant avec la méthode `bindParam` :
    ↪ Possibilité de spécifier en plus des types, tailles, *etc.*
    ↪ Évite les injections SQL

```
$requete->bindParam(':nom', 'Schwarzenegger', PDO::PARAM_STR, 100);
if($requete->execute()) {
  ...
}
```

Autres méthodes de `PDOStatement`

- `columnCount` : nombre de colonnes dans le résultat
- `rowCount` : nombre de lignes affectées lors de la dernière requête
  ↪ Utile pour vérifier les ajouts, suppressions, *etc.*
- *etc.*

Généralités

- Peut être utilisé via une classe (comme PDO) :
  ↪ Classes `mysqli`, `mysqli_result`, `mysqli_stmt`
- Possible aussi d'utiliser les fonctions `mysqli_*`

Exemples

Méthodes	Fonctions
`mysqli::__construct`	`connect`
`mysqli::query`	`mysqli_query`
`mysqli_result::fetch_array`	`mysqli_fetch_array`

Dans un souci de cohérence, il est préférable d'utiliser les classes

Connexion à une base de données

```
define("BD_HOST", "localhost");
define("BD_BASE", "info0303");
define("BD_USER", "root");
define("BD_PASSWORD", "");

$BD = new mysqli(BD_HOST, BD_USER, BD_PASSWORD, BD_BASE);
if ($BD->connect_errno)
    echo "<p> Problème de connexion à la base de données (".
        $BD->connect_errno.") ".$BD->connect_error."</p>";
else
    echo "<p> Connexion à la base OK. </p>";
$BD->set_charset("utf8");
```

- À noter que *localhost* $\neq$ `127.0.0.1` pour *mysqli*
- Ne pas oublier `set_charset` pour spécifier le jeu de caractères

Sélection (1/2)

```
$SQL = <<<SQL
  SELECT `etu_nom`, `etu_prenom`, `grp_intitule`
  FROM `etudiant`, `groupe` WHERE `etu_groupe`=`grp_id`;
SQL;

if($requete = $BD->query($SQL)) {
    echo "<ul>";
    while($ligne = $requete->fetch_assoc())
      echo <<<HTML
<li><b>{$ligne['etu_nom']} {$ligne['etu_prenom']}</b> :
       {$ligne['grp_intitule']} </li>
HTML;
    echo "</ul>";
}
else
    echo "<p>␣Erreur␣lors␣de␣l'exécution␣de␣la␣requête.␣</p>";
```

- `query` retourne un objet `mysqli_result`

Sélection (2/2)

- La méthode `fetch_array` retourne un tableau associatif :
  ↪ Par défaut, données associées au nom de colonne + au numéro
  ↪ `[ 'etu_nom' => "Schwarzenegger", 0 => "Schwarzenegger", ...]`
- Possible de spécifier d'autres modes :
  - `MYSQLI_BOTH` : par défaut
  - `MYSQLI_ASSOC` : valeurs indexées par le nom de la colonne
    ↪ Ou méthode `fetch_assoc`
  - `MYSQLI_NUM` : valeurs indexées par le numéro de la colonne
    ↪ Ou méthode `fetch_row`

Requêtes préparées : get_result

```
$SQL = <<<SQL
SELECT `etu_nom`, `etu_prenom`, `grp_intitule` FROM `etudiant`, `groupe`
WHERE `etu_groupe`=`grp_id` AND `etu_nom`= ?;
SQL;

if($requete = $BD->prepare($SQL)) {
  $requete->bind_param("s", $nom);
  $nom = "Schwarzenegger";
  if($requete->execute()) {
    $resultat = $requete->get_result();
    echo "<ul>";
    while($ligne = $resultat->fetch_assoc())
      echo "<li><b>{$ligne['etu_nom']} {$ligne['etu_prenom']}</b> : ".
           "{$ligne['grp_intitule']}</li>";
    echo "</ul>";
 }
 else
   echo "<p> Erreur lors de l'exécution de la requête. </p>";
}
else
  echo "<p> Erreur lors de la préparation de la requête. </p>";
```

Requêtes préparées : `bind_result`

```
$SQL = <<<SQL
  SELECT `etu_nom`, `etu_prenom`, `grp_intitule`
  FROM `etudiant`, `groupe`
  WHERE `etu_groupe`=`grp_id` AND `etu_nom`= ?;
SQL;
if($requete = $BD->prepare($SQL)) {
    $requete->bind_param('s', $nom);
    $nom = 'Schwarzenegger';
    if($requete->execute()) {
        $requete->bind_result($nom, $prenom, $groupe);
        echo "<ul>";
        while($resultat = $requete->fetch())
            echo "<li> <b>$nom $prenom</b> : $groupe </li>";
        echo "</ul>";
    }
    else
        echo "<p> Erreur lors de l'exécution de la requête. </p>";
}
else
    echo "<p> Erreur lors de la préparation de la requête. </p>";
```

Conclusion

- La plupart des applications PHP utilisent une BD
- Vous pouvez utiliser au choix PDO ou *mysqli*
- Différentes étapes :
  1. Analyser le problème
  2. Trouvez les données nécessaires
  3. Conception de la base de données
  4. Écriture du code PHP
- Une partie sera vue en INFO0304 (Bases de données)
  ↪ Analyse, conception, SQL
- Une autre partie sera réalisée avec *Laravel*
  ↪ Nous reviendrons sur les bases de données dans le cours sur *Laravel*