

Les bases de données

Cyril Rabat

`http://cyril-rabat.fr`

Programmation Web - Php

2017-2018

Cours Info 303
Bases de données et PHP

Version 22 octobre 2018

Table des matières

- 1 Conception d'une base de données
 - Quelques rappels
- 2 Création et manipulations d'une base de données
 - Introduction
 - Définition d'une base de données
 - Manipulation des données
- 3 Manipulation de bases de données en PHP
 - Introduction
 - PDO
 - *mysqli*

À quoi sert une base de données ?

- Permet de stocker des données indexées :
↪ ≠ 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 :
 - ↪ Langage SQL (*Structured Query Language*)
 - ↪ Création, recherches, combinaisons, tris, *etc.*

Conception d'une base de données

- Le cycle de vie d'une base de données :
 - Conception : schéma conceptuel
 - Implantation des données
 - Utilisation des données
 - Maintenance de la base
- Pour concevoir une base de données :
 - Expression des besoins
 - ↪ Indépendant de la solution retenue (SGBD + schéma de la base)
 - Projection sur une solution
 - ↪ Permet l'implantation

Le langage SQL

- SQL pour *Structured Query Language*
- Langage utilisé pour interagir avec les SGBD :
 - ↪ Création, ajout de données, requêtes, etc.
- Standardisé
 - ↪ Mais différentes versions !
 - ↪ Syntaxe variable suivant les SGBD

Comment créer une base de données ?

- Connexion à un SGBD puis saisie des commandes via un CLI
- Utilisation d'une interface graphique
 - ↔ Exemple : *phpMyAdmin*
- Avantage de *phpMyAdmin* :
 - Importation/exportation de la base (structures et/ou données)
 - Création/modification des tables à la souris
 - Possible d'exécuter des requêtes SQL
 - Outils divers, dont le concepteur
- Avec une distribution comme *Wamp* ou *EasyPHP* :
 - ↔ Accès à la base : `http://localhost/phpmyadmin/`
 - ↔ Par défaut, connexion : login = root ; mot de passe = aucun !

Première base de données

- Avec *phpMyAdmin* :
 - Nouvelle base de données
 - ↪ Nom : INFO303
 - ↪ Interclassement : `utf8_general_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 (par exemple)
 - En SQL :
 - Sélection de la base
 - Onglet SQL
 - Saisie de la requête

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 ressource, plus lent

Exemple : retour sur les auditeurs

- Un auditeur est associé à un centre
- Auditeur (id, nom, prenom, centre)
 - Clé primaire : id
 - Clé étrangère : centre qui référence Centre(id)
- Centre (id, nom, adresse)
 - Clé primaire : id

Types des données

- Auditeur :
 - id : entier
↪ `unsigned int`
 - nom, prénom : chaînes de caractères
↪ `varchar(100)`
 - centre : entier
↪ `unsigned int`
- Centre :
 - id : entier
↪ `unsigned int`
 - Nom : chaîne de caractères
↪ `varchar(100)`
 - Adresse : texte (longueur variable)
↪ `text`

Création d'une table

```
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 ...

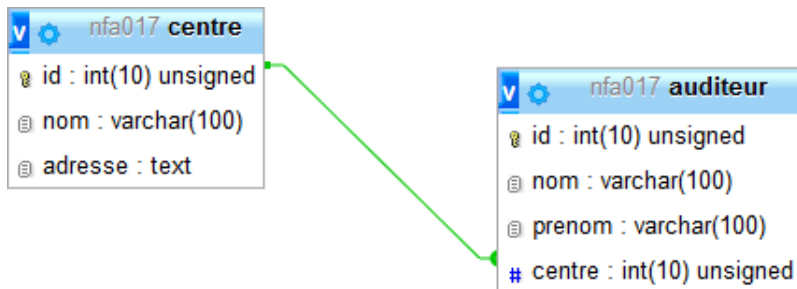
↪ VARCHAR(taille), TEXT ...

↪ DATE, TIME, ...

Création des tables auditeur et centre (1/2)

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

Création des tables auditeur et centre (1/2)



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 à une table
- Plusieurs enregistrements possibles (séparés par ',')

```
INSERT INTO `centre` (`id`, `nom`, `adresse`) VALUES  
(NULL, "Grand_Est",  
 "4_rue_du_Dr_Heydenreich_CS_65228_54052_Nancy_cedex"),  
(NULL, "Bourgogne_Franche-Comté",  
 "ENSMM_-_26_chemin_de_l'Epitaphe_25030_Besançon");  
  
INSERT INTO `auditeur` (`id`, `nom`, `prenom`, `centre`) 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 supprimés !

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

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 centre des auditeurs du centre 2 */  
UPDATE `auditeur` SET `centre`=1 WHERE `centre`=2
```


Sélection de données (1/3)

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

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

Exemple

```
SELECT * FROM `auditeur`
```

Résultat :

id	nom	prenom	centre
1	Schwarzenegger	Arnold	1
2	Eastwood	Clint	2
3	Stallone	Sylvester	1

Sélection de données (2/3)

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

- Possible de spécifier des champs/colonnes spécifiques

Exemple

```
SELECT `nom`, `prenom` FROM `auditeur`
```

Résultat :

nom	prenom
Schwarzenegger	Arnold
Eastwood	Clint
Stallone	Sylvester

Sélection de données (3/3)

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

- Sélectionne des données de plusieurs tables
↪ Jointure

Exemple : provoque une erreur

```
SELECT `nom`, `prenom` FROM `auditeur`, `centre` WHERE `centre`=`id`
```

Ambiguïté sur les champs 'id' et 'nom' :
présents dans les tables 'auditeur' et 'centre'.



Sélection de données (3/3)

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

- Sélectionne des données de plusieurs tables
↪ Jointure

Solution : spécifier le nom des tables et/ou les alias

```
SELECT A.`nom`, A.`prenom`, B.`nom` AS `centre`  
FROM `auditeur` AS A, `centre` AS B WHERE A.`centre`=B.`id`
```

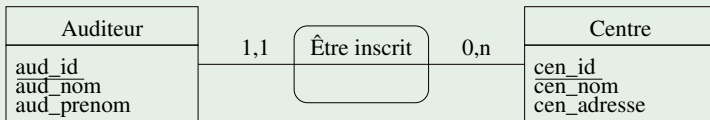
Résultat :

nom	prenom	centre
Schwarzenegger	Arnold	Grand Est
Eastwood	Clint	Bourgogne Franche-Comté
Stallone	Sylvester	Grand Est

Remarques

- Les requêtes peuvent devenir très vite complexes !
- Solutions :
 - Réfléchir au préalable à la structure de la base
↪ Au moment de la définition du MCD
 - Utiliser une notation spécifique
↪ Utilisation de préfixes

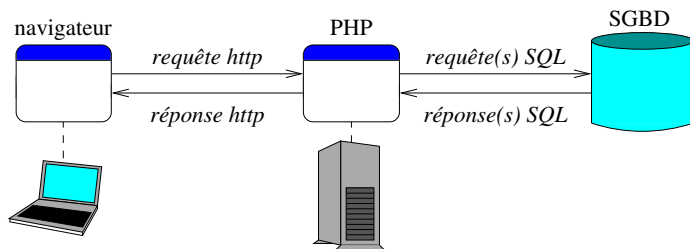
Exemple



```
SELECT `aud_nom`, `aud_prenom`, `cen_nom`  
FROM `auditeur`, `centre` WHERE `aud_centre`=`cen_id`
```

Introduction

- Base de données utilisée pour stocker des données
- 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 :
 - API indépendante du SGBD
 - Simple, portable
 - Inconvénients :
 - Ne permet pas de profiter pleinement des fonctionnalités de MySQL
- *mysqli* :
 - API spécifique à MySQL
 - Recommandée pour l'utilisation de MySQL version 4.7 et plus
 - ↪ Version fournie par Wamp : 5.7.14 (au 20/12/2017)

Attention

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

Connexion à une base de données

```
define("BD_HOST", "localhost");
define("BD_BASE", "info303");
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>";
    exit();
}
```

- 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

Sélection (1/2)

```
$SQL = "SELECT `aud_nom`, `aud_prenom`, `cen_nom`, `cen_adresse`  
FROM `auditeur`, `centre`  
WHERE `aud_centre` = `cen_id`";  
  
if($requete = $BD->query($SQL)) {  
    echo "<ul>";  
    while($resultat = $requete->fetch()) {  
        echo "<li><b>". $resultat["aud_nom"]. "  
            $resultat["aud_prenom"]. "</b>: ". $resultat['cen_nom'].  
            " (" . $resultat['cen_adresse'] . ")</li>";  
    }  
    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
 - ↪ `['aud_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 = "INSERT INTO `Auditeur`".  
      "(`aud_id`,`aud_nom`,`aud_prenom`,`aud_centre`)".  
      "VALUES (NULL,`De_Niro`,`Robert`,2)";  
  
if($requete = $BD->exec($SQL))  
    echo "<p>$_requete_ligne(s) ajoutée(s).</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 :
 - Utile lorsque des requêtes sont exécutées plusieurs fois
 - Évite du temps de préparation à la base
 - Limite les problèmes d'injection
- L'exécution de la requête est réalisée en deux temps :
 - Préparation de la requête
 - Exécution de la requête à partir de paramètres
↪ Plusieurs fois si nécessaire
- Utilisation de la méthode `PDO::prepare`
↪ Possible d'ajouter des paramètres
- Exécution avec `PDOStatement::execute`

Requêtes préparées (2/2)

```
$SQL = "SELECT `aud_nom`, `aud_prenom`, `cen_nom`, `cen_adresse`".
        "FROM `auditeur`, `centre`".
        "WHERE `aud_centre`=`cen_id` AND `aud_nom`=:nom;";

if($requete = $BD->prepare($SQL)) {
    if($requete->execute(array(':nom' => "Schwarzenegger"))) {
        echo "<ul>";
        while($res = $requete->fetch(PDO::FETCH_ASSOC)) {
            echo "<li><b>". $res["aud_nom"]. " ". $res["aud_prenom"]. "</b> ".
                $res["cen_nom"]. " (" . $res["cen_adresse"] . ") </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>";
```

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
- 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
- *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
<code>mysqli::__construct</code>	<code>connect</code>
<code>mysqli::query</code>	<code>mysqli_query</code>
<code>mysqli_result::fetch_array</code>	<code>mysqli_fetch_array</code>

Connexion à une base de données

```
define("BD_HOST", "localhost");
define("BD_BASE", "info303");
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 = "SELECT `aud_nom`,`aud_prenom`,`cen_nom`,`cen_adresse`  
        `FROM` `auditeur`,`centre` WHERE `aud_centre`=`cen_id`";  
  
if($requete = $BD->query($SQL)) {  
    echo "<ul>";  
    while($resultat = $requete->fetch_assoc()) {  
        echo "<li><b>".$resultat["aud_nom"]." ".  
            $resultat["aud_prenom"]."</b>:".$resultat["cen_nom"].  
            " (".$resultat["cen_adresse"].")</li>";  
    }  
    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
 - ↪ `['aut_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

```
$SQL = "SELECT `aud_nom`,`aud_prenom`,`cen_nom`,`cen_adresse`".  
        "FROM `auditeur`,`centre`".  
        "WHERE `aud_centre`=`cen_id` AND `aud_nom`=_?";  
if($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["aud_nom"]. " ". $ligne["aud_prenom"].  
                "</b>_ ". $ligne["cen_nom"]. " (" . $ligne["cen_adresse"].  
                ")</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 = "SELECT `aud_nom`,`aud_prenom`,`cen_nom`,`cen_adresse`".
        "FROM `auditeur`,`centre`".
        "WHERE `aud_centre`=`cen_id` AND `aud_nom`=?";

if($requete = $BD->prepare($SQL)) {
    $requete->bind_param("s", $nom);
    $nom = "Schwarzenegger";
    if($requete->execute()) {
        $requete->bind_result($nomAuditeur, $prenomAuditeur,
                               $nomCentre, $adresseCentre);

        echo "<ul>";
        while($resultat = $requete->fetch()) {
            echo "<li><b>$nomAuditeur_$prenomAuditeur</b>:".
                "$nomCentre_($adresseCentre)</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>";
```