Développement Web avancé NFA017

Fichiers, Base de données et php

J-F Berger

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

Fichiers (1)

Sous réserve des droits de lecture écriture, on peut écrire et/ou lire des fichiers enregistrés sur le serveur

fopen("nom du fichier","mode") renvoie le descripteur du fichier et ouvre un fichier pour une opération de lecture ou écriture

- " r " : lecture uniquement, au début du fichier
- " w " : écriture uniquement, création du fichier si nécessaire,(sinon supprime le contenu précédent), au début du fichier
- " a " : ajout au fichier(écriture seule), création du fichier si nécessaire, commence à la fin du fichier
- " r+ " : lecture et écriture, création du fichier si nécessaire, au début du fichier
- "w+ " : lecture et écriture, création du fichier si nécessaire, supprime le contenu précédent
- "a+ " : lecture et écriture, création du fichier si nécessaire, commence à la fin du fichier

- si le nom de fichier commence par http: // une connexion http1.x vers le serveur specifié est ouverte
  si le nom de fichier commence par ftp:// une connexion ftp vers le serveur specifié est ouverte (mode ftp passif , en lecture seulement ou en écriture seulement)
  dans les autres cas, fichier local au serveur

Fichiers (2)

- **close($descripteur)** ferme le fichier
- $chaine = **fread($descripteur,$length)**
  lit jusqu'à $length dans le fichier référencé par $descripteur (ou juqu'à la fin du fichier si filesize("nom du fichier")<$length)
- **fwrite($descripteur,$chaine,$length) ou fputs(...)**
  écrit dans le fichier référencé par $descripteur le contenu de la chaîne $chaine
  si $length est fourni, l'écriture s'arrête après $length octets ou à la fin de la chaîne (1ère condition remplie)
- **is_dir($nomf)** : renvoie TRUE si $nomf est un répertoire
- **is_file($nomf) :** renvoie TRUE si $nomf est un fichier

Fichiers (3)

- **mkdir("nom_chenim,$mode_octal)** : créé un répertoire selon le chemin précisé, avec le mode défini (0777 à 0000) en octal.

le propriétaire est le propriétaire du processus httpd

- **rmdir($dirname) :** suppression d'un répertoire
- **chmod($descripteur,$newmodeoctal)** permet de modifier le mode
- **file_exists**("nom du fichier")
- **chown, chgrp** : changement du owner , du groupe
- **copy($desc_source_name,$desc_dest_name)** : copie de source vers destination
- **unlink($filename) :** suppression fichier

autres outils de base

- **file_get_contents(‘url du fichier’) :** pour récupérer l'integralité du fichier. (pas d’open!,ni descripteur)
- **fgetc ($descripteur)** est utilisé pour lire les caractères dans un fichier un par un (grâce à un pointeur)
  - **rewint($descripteur);** pour se placer en début de fichier
  - **fseek($descripteur , $offset);** déplace le pointeur interne du fichier de $offset (Attention, cette fonction retourne 0 en cas de succès, et sinon -1)
  - **ftell($descripteur);** renvoie la position actuelle du pointeur
- **fgets($descripteur, $lon); :** pour cette fonction la lecture s'arrête à la lecture d'un caractère de fin de ligne**.**
- **file($descripteur)** retourne le fichier dans un tableau. Chaque élément du tableau correspond à une ligne du fichier, et les retour chariots sont placés en fin de ligne**.**

Introduction aux bases de données

Introduction aux bases de données et à SQL

- Bases de données:
  - stockage de données , ordonnées en unités hiérarchisées
  - Une base de données:
    - Des tables (nom) qui contiennent des enregistrements (lignes)
    - Un enregistrement : composé de plusieurs champs (colonnes)
    - Chaque champ a un nom et un type de données (texte, nombre,date,heures, blob (Binary Large Object pour les images)
  - Les tables sont reliés entre elles par un **schéma de relation ou "vue conceptuelle" qui décrit la structure (relationnelle) de la base**
- **SQL: structured Query Language** : Créer, extraire, modifier, supprimer, fusionner …
  - Cad communiquer avec des bases de données relationnelles dotées d' un SGBDR
  - Apparu en 1979, standardisé ANSI (v3 en 1999)

exemple

- Table personnes

table

Champs

type

personnes			
Id	**Nom**	**Prénom**	**CP**
number(10)	**varchar(20)**	**varchar(20)**	**number(5)**

Id: clé unique (clé primaire) pour accès rapide non équivoque à un enregistrement

- Table Code_postal

Code_postal	
nom_ville	**CP**
varchar(20)	**Number(5)**

SQL

- **SQL : non procédural**
  - **On indique ce qu'on cherche , le SGBD décide comment récupérer l'information**
- **La manipulation des données** :
  - requêtes regroupant des instructions SQL
  - Une requête est une question (avec conditions)

  Si la BD comporte des données vérifiant la condition, SQL retourne les données.
- PHP permet d'incorporer des requêtes dans un programme et de transmettre ces requêtes au SGBD par des **fonctions php.**
  - La plupart des SGBDR accessibles (Oracle, Sybase, Microsoft SQL Server, PostgreSQL , MySQL,….)
  - Sqlite :petite bibliothèque écrite en C qui propose un moteur de base de données SQL , a été intégrée dans php5

Détails PHP/SGBD (1)

La connexion : fonction php

mysql_connect("nom_serveur","nom_utilisateur","mot_passe"**)**;

mssql_connect("nom_serveur,"nom_utilisateur","mot_passe"**)**;

oci_connect("nom_utilisateur","mot_passe", "nom_base"**)**;

pg_connect("dbname=nom_base host=nom_serveur port=num_port user=nom_utilisateur password=mot_passe"**)**;

sybase_connect("nom_serveur","nom_utilisateur","mot_passe"**)**;

Qui retourne un **$id_connexion**;

- PDO fournit une interface d'abstraction à l'accès de données, ce qui signifie que vous utilisez les mêmes fonctions pour exécuter des requêtes ou récupérer les données quelque soit la base de données utilisée.

Il existe deux façons de se connecter à une base de données.

- Les connexions **non-persistantes** (*base_**connect***). (se ferment automatiquement à la fin du script créateur)

Les connexions **persistantes** (*base_**pconnect***).réutilisables

donc réutilisables sans ouverture (bon moyen d'accélérer les accès à une base ,mais le nombre maximal de connexion risque d'être atteint rapidement…) sinon pas de fonctionnalités différentes

Détails PHP/SGBD (2)

- **Sélection** d' une base de données

  mysql_select_db($nom_base_donnee, $id_connexion**);**

  sybase_select_db($nom_base_donnee, $id_connexion**);**

  (faite dans le pg_connect et oci_connect)

- **Création** de base de données (ou par sql)

  mssql_create_db($nom_base_donnee, $id_connexion**);**

  mysql_create_db ($nom_base_donnee, $id_connexion**);**

- **Suppression** d' une base de données (ou par sql)

  mssql_drop_db($nom_base_donnee, $id_connexion**);**

  mysql_drop_db ($nom_base_donnee, $id_connexion**);**

- **Utilisation de SQL:**

$requete = "CREATE DATABASE nom_base_donnee";

$pg_exec($id_connexion,$requete);

// il faut que l'utilisateur (dans la création de $id_connexion) ait les droits de créer une base!!

Sql DDL (Data Definition Language)

```
CREATE TABLE "Fiche_Personne "  (
 ID NUMBER(8) UNIQUE,
 Sexe VARCHAR(50) NOT NULL,
 Nom VARCHAR(50) NOT NULL,
 Prenom VARCHAR(50) NOT NULL,
 Snd_Prenom VARCHAR(50) NULL,
 Adresse VARCHAR(100) NOT NULL,
 Code_Postal NUMBER(5) NOT NULL,
 Ville VARCHAR(30) NOT NULL,
Telephone NUMBER(10) NULL,
 eMail VARCHAR(30) NULL)
```

Création d'une table
Avec contraintes sur certains champs :
ID Unique
sexe, nom, prenom obligatoires
….

CONTRAINTES

NULL indique que la colonne peut NE PAS être renseignée.
NOT NULL indique que la colonne DOIT contenir une valeur.
PRIMARY KEY indique que la colonne constitue la clé primaire de la table.
UNIQUE impose que chaque valeur de la colonne doit être unique.

Sql DDL (Data Definition Language) 2

- D'autres fonctions:
  - ALTER TABLE pour supprimer, changer la définition des colonnes ou les contraintes
  - DROP TABLE pour supprimer une table (avec variantes RESTRICT(erreur si référencement par une vue) ou CASCADE
  - CREATE INDEX : pour créer un index référençant une ou plusieurs colonnes (performance)
  - DROP INDEX
  - CREATE VIEW permet de créer une table virtuelle par une requête de sélection
    - Utilisable comme une table
    - Pas de stockage physique
    - Permet de ne donner accès qu'aux données définies dans la vue (sécurité)

CREATE VIEW fiche_femme **AS** SELECT nom, prenom, adresse, cp, ville FROM tbl_personnel WHERE sexe = 'F'

Sql DQL (Data Query Language)

- **SELECT * | { [ALL | DISTINCT] nom_colonne,...,nom_colonneN } FROM nom_table WHERE Condition**
  - Sélectionne un à plusieurs champs (distincts si DISTINCT)

  FROM l'extraction provient de une ou plusieurs tables

  WHERE la condition booléenne est vérifiée

SELECT **nom_table**.nom_colonne,..., **nom_tableN**.nom_colonneN F**ROM nom_table,..., nom_tableN**

WHERE nom_tableN.nom_colonneJ='F'

ORDER by nom_table.nom_colonne

L'**expression conditionnelle** peut être construite à l'aide de **plusieurs prédicats** constitués d'**opérateurs de comparaisons** (= | <> | != | > | > = | !> | < | <= | !<) et séparées par des **opérateurs booléens** *AND*, *OR* ou *NOT*.

Sql DQL (Restriction sur une comparaison de chaîne)

- Le prédicat *LIKE* permet de faire des comparaisons sur des chaines grâce à des caractères, appelés caractères *jokers* :
- Le caractère **%** permet de remplacer une séquence de caractères (éventuellement nulle)
- La caractère _ permet de remplacer un caractère (l'équivalent du "blanc" au scrabble...)
- Les caractères **[-]** permettent de définir un intervalle de caractères (par exemple [J-M])

- Exemple :
  La sélection des contacts dont le nom a un E en deuxième position se fait par l'instruction :

```
SELECT * FROM AGENDA WHERE nom LIKE "_E%"
```

SQL DML (Data manipulation L)

- UPDATE : La commande *UPDATE* permet de mettre à jour des données existantes au sein d'une table.

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

- INSERT INTO: La commande *INSERT INTO* permet d'ajouter des données dans une table.

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

- DELETE FROM :La commande *DELETE FROM* permet de supprimer des enregistrements au sein d'une table.

  DELETE FROM Nom_Table [WHERE Condition]

  PS : si pas de condition, tous les enregistrements sont supprimés…

On peut utiliser une sous-requête pour exprimer la condition :

DELETE FROM fiche_personnes WHERE ville IN (SELECT ville FROM table_ville)

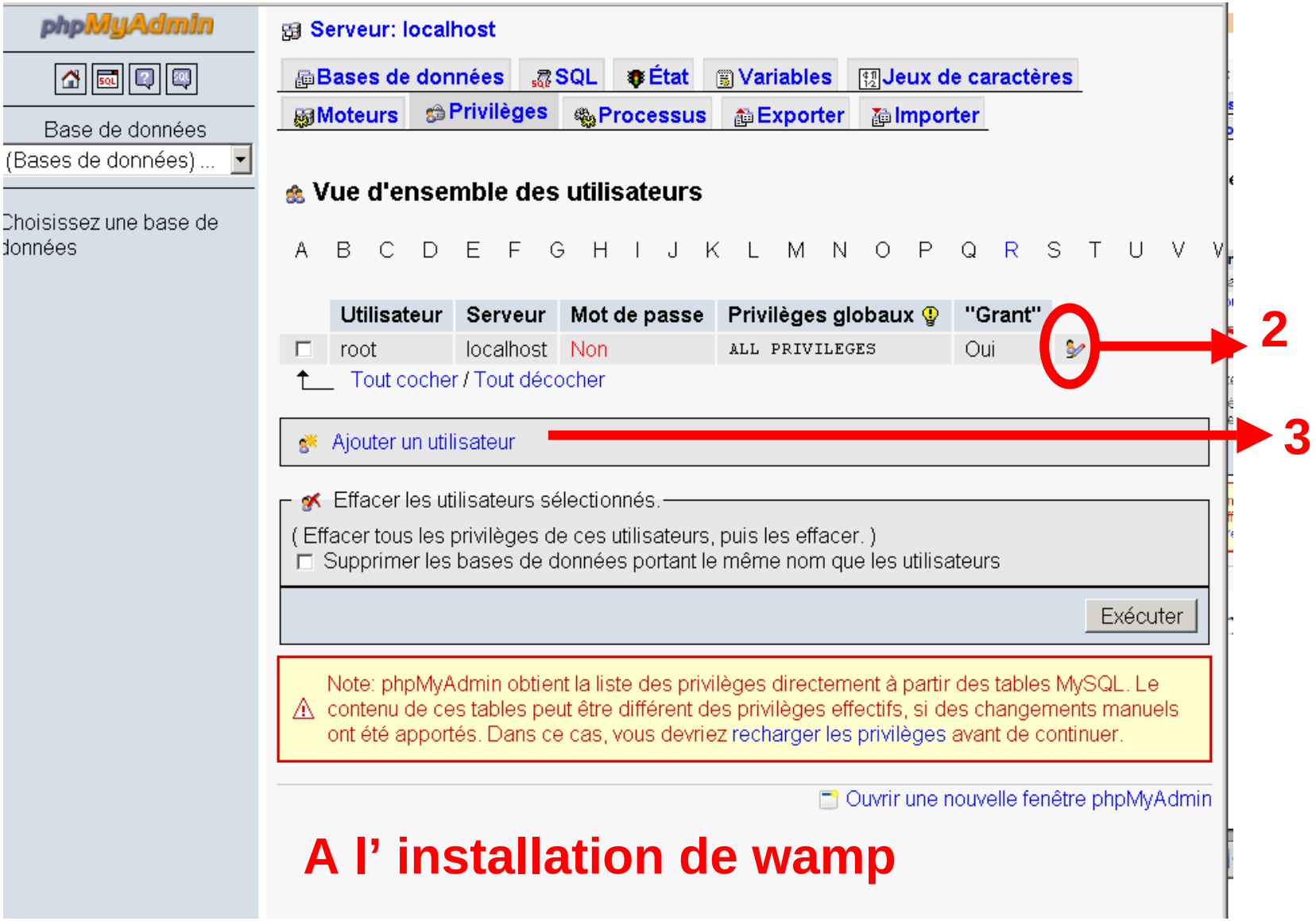
SQL DCL (Data Control Language) (1)

- Pour gérer les droits d'accès aux objets de la base
  - **Un utilisateur de base de données doit posséder un compte de sécurité** afin d'accéder aux tables ou à tout autre élément, ou d'obtenir des droits d'exécution de requêtes SQL.
  - Pour cela, il faut non seulement que l'utilisateur soit créé mais également que des privilèges lui soient accordés.
- ***L'administration des utilisateurs est liée à l'implémentation***

- ***Exemple sous postgresql:***
  - **Ligne de commande**
    - **L'administrateur crée des utilisateurs par CREATE USER**
    - **ALTER USER permet de modifier ceux-ci**
    - **GRANT et REVOKE permettent de gérer leurs droits**
  - **– phpPgAdmin : interface pour la gestion directe de ces propriétés…**

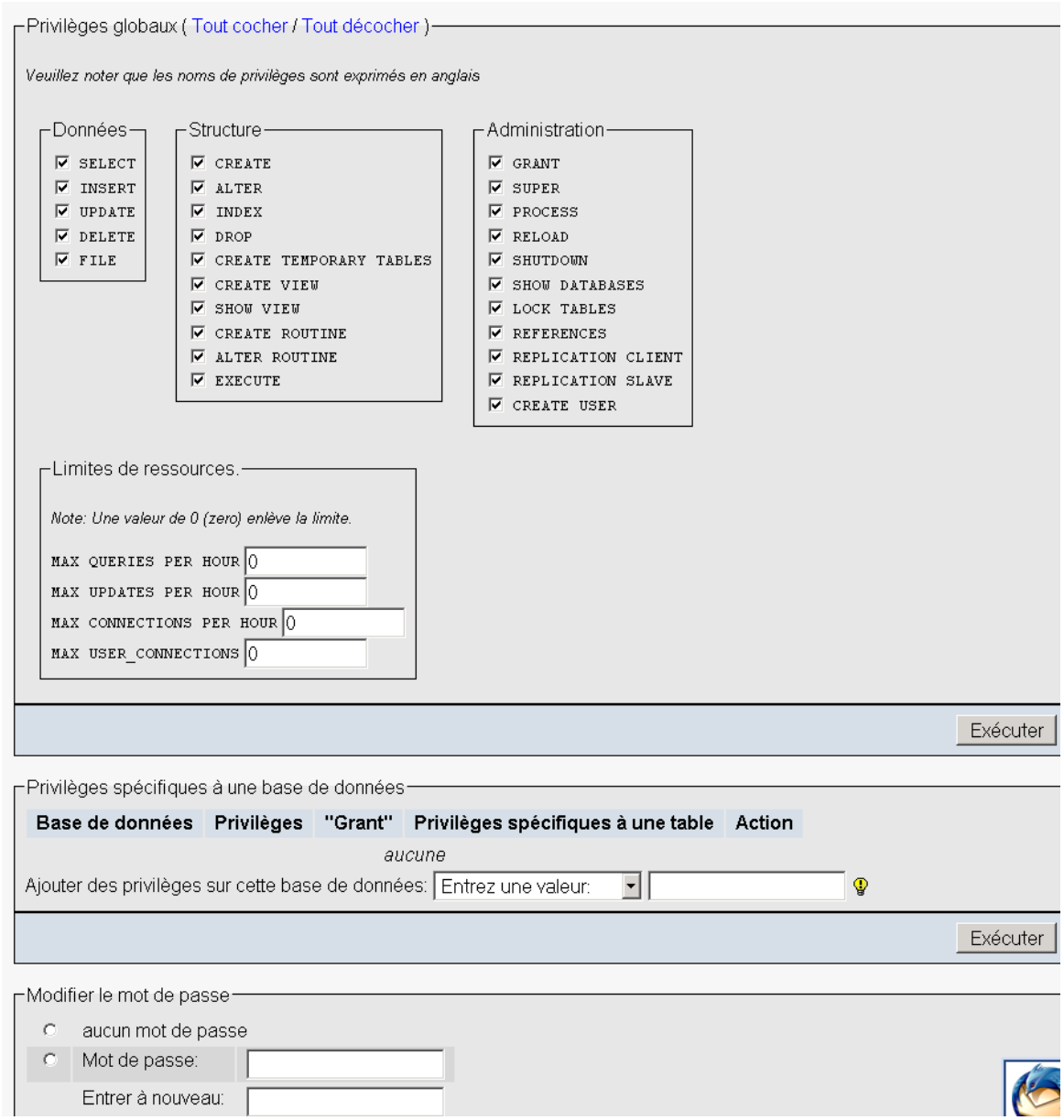
si on en a les droits…

phpMyadmin (1)



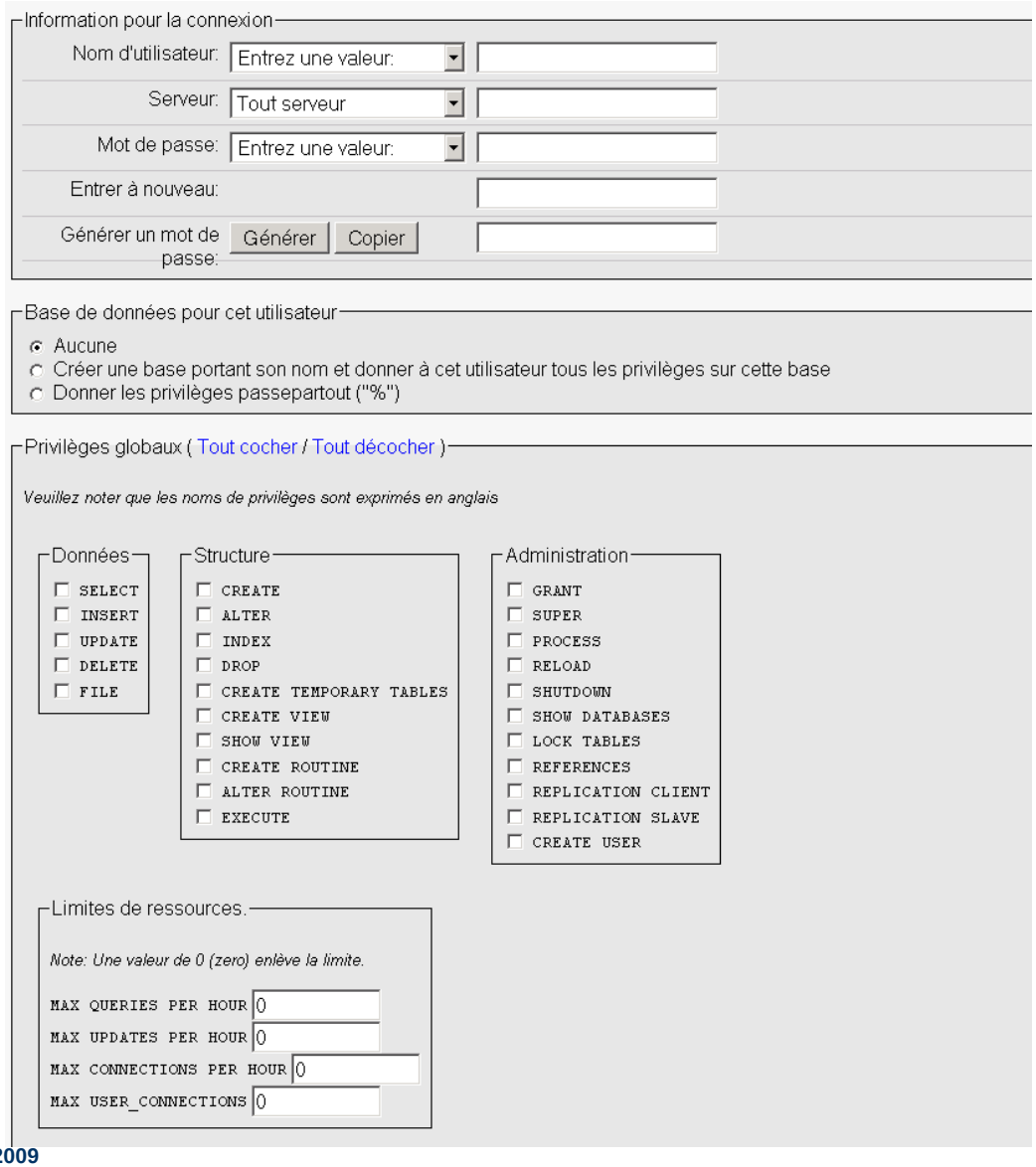
A l' installation de wamp

Exemple: modification user root (2)



Modification des droits de l'administrateur
Et du mot de passe

Exemple: création utilisateur (3)



Création d'un utilisateur
et de ses droits
lecteur_pur

SQL DCL (Data Control Language) (2)

- ***Sous mysql,***
  - **on dispose d'une base utilisateur mysql, installée à l'utilisation qui gère les utilisateurs et leurs privilèges d'accès sur l'ensemble des bases de données, y compris mysql.**
    C'est l'utilisateur privilégié "root" qui est en charge d'administration de cette base. C'est donc le seul qui pourra ajouter des utilisateurs, en retirer, et modifier leurs droits.
  - D'où l'urgence a définir un mot de passe pour cet utilisateur root
- La **base mysql** (tables user,db,host,tables_priv,columns_priv)
  - La table **user** recense tous les utilisateurs qui ont accès au serveur des bases de données MySQL.
    Elle contient les privilèges de chacun d'entre eux dans tout mysql**.**
  - La table **db** permet de lier un utilisateur à une ou plusieurs base(s).
  - La table **host** permet de lier un hôte à une base de données.

SQL Data Control Language (3)

Toujours sous MySQL, pour créer un utilisateur et lui affecter des droits on peut donc directement travailler sur ces tables.

- Il est cependant conseillé de travailler avec les requêtes **grant** et **revoke** : Les commandes GRANT et REVOKE permettent à l'administrateur système de créer et supprimer des comptes utilisateur et de leur donner ou retirer des droits.

Les droits sont donnés à 4 niveaux :

- **Niveau global** :Les droits globaux s'appliquent à toutes les bases de données d'un serveur. Ces droits sont stockés dans la table mysql.user.

  *Exemple :REVOKE ALL ON *.* retirera seulement les privilèges globaux.*

- **Niveau base de données** **Les droits de niveau de base de données s'appliquent à toutes les tables d'une base de données. Ces droits sont stockés dans les tables mysql.db et mysql.host**.

  Exemple REVOKE ALL ON db. retirera seulement les privilèges de base de données*

- **Niveau table :** Les droits de table s'appliquent à toutes les colonnes d'une table. Ces droits sont stockés dans la table mysql.tables_priv.

  REVOKE ALL ON db.table retirera seulement les privilèges de table.

- **Niveau colonne** : Les droits de niveau de colonnes s'appliquent à des colonnes dans une table. Ces droits sont stockés dans la table mysql.columns_priv.

SQL et PHP

- **Le langage PHP dispose de nombreuses fonctions permettant de travailler sur des bases de données MySQL.**
- L'utilisation de ces fonctions nécessite une **compilation de PHP avec le support MySQL en activant l'option**

--with-mysql=[répertoire].

Le répertoire indiqué correspond au chemin des librairies du SGBDR MySQL.

Pour posgresql : **option*--with-pgsql[=répertoire]***

- **Wamp et Easyphp** : par défaut **MySQL et phpmyadmin** (outil web d'administration BD)

Requêtes SQL : les bases

- **Voici les étapes nécessaires pour passer une requête SELECT, (détails dans la suite) :**
  - **Connexion au serveur MySQL** (authentification) : nous récupérons une *ressource de connexion* sur laquelle nous pourrons travailler.
  - **Sélection d'une base de données**
  - **Passage de la requête** : nous récupérons ici une *ressource*, qui contient (mais pas de manière immédiatement accessible) la réponse à notre requête.
  - **Vérification de la validité de la ressource récupérée** : pour s'assurer que la réponse du serveur est valide.
  - **Extraction des données** à partir de la ressource : pour récupérer les informations dans un format exploitable par PHP.
  - **Fermeture de la connexion**.

MySql : Connexion et sélection

```
<?php
// Connexion à MySQL
   $link = mysql_connect("mysql_hote", "mysql_user",
   "mysql_pw")
            or die("Impossible de se connecter");

   echo "Connexion à MySql réussie";

// Sélection d'une BD
   mysql_select_db("ma_BD") or die("sélection ma_BD
   impossible");

    echo "Connexion à la Base de Donnée 'ma_BD'
   réussie";
?>
```

Les fct MySql : Création d'une BD

```
<?php
//Création d'une BD
$sql = 'CREATE DATABASE ma_nouvelle_BD';
if (mysql_query($sql, $link))    {
     echo "Base de données créée correctement<p />";
}else{
      echo 'Erreur lors de la création de la base de données : ‘;
      echo  mysql_error() . "<p />";
}
?>
```

On peut aussi utiliser la fonction php :

bool **mysql_create_db** (‘ma_nouvelle_BD’,$link); *qui retourne TRUE or FALSE*

SQL query (select) (1)

- **Passage d'une requête SELECT et récupération d'une ressource**

```
$requete=« SELECT nom, service FROM employes
   WHERE
statut='stagiaire' ORDER BY nom »;
$result=mysql_query($requete,$id_connexion) or
die ($requete. " - " . mysql_error());
```

  - Retourne une ressource *pour une requête SELECT*.
  - Retourne true ou false pour les autres requêtes (*UPDATE*, *DELETE*, *DROP*, etc…)

le "or die" affiche l'erreur MySQL en cas d'échec de la requête,

SQL query(2) Exploitation de la ressource

- **Nombre d'enregistrements retournés pour un SELECT:**
  $nb_result=mysql_num_rows ($result)
- **Nombre de lignes affectées** par les requêtes du type *DELETE*, *INSERT*,

REPLACE, ou *UPDATE :*

$nb_affect= mysql_affected_rows($result)

- **Récupération des résultats (M1)**
  - **Tableau : mysql_fetch_array ($result)**
  - extrait un enregistrement (et un seul), une ligne de la table et déplace le pointeur.
  - Retourne false si plus de ligne

Exemple :

```
$result = mysql_query("SELECT id, nom FROM fiche_personne");
while ($row = mysql_fetch_array($result)) {
    printf("ID : %s  Nom : %s", $row[0], $row[1]);
}
```

SQL query (3) exemple 2 de base M1

- **Exemple 2 (1ère méthode)**

```
$link = mysql_connect("localhost", "cours", "passw");
$query = "SELECT * FROM table_test";
$result = mysql_query($query, $link) or die($query . " - " .
    mysql_error());
$nbResults = mysql_num_rows($result);
echo " on a $nbResults enregistrements   <br/> "  ;
//affichage des champs id et comment de la table
    table_test (tableau)
while ($tab = mysql_fetch_array($result)){
    echo $tab['id'] . " : " . $tab['comment']; echo "<br />";
}
```

SQL query M2

- **Récupération des résultats (2 ème méthode)**
  - **mysql_fetch_assoc ($result)**
    - Retourne un tableau associatif correspondant à la ligne récupérée et déplace le pointeur.
    - Retourne false si plus de ligne

Exemple M2:

```
$result = mysql_query("SELECT id, nom FROM
    fiche_personne");
while ($row = mysql_fetch_assoc($result)) {
        echo $row["id"];
        echo "  ";
        echo $row["nom"];
}
```

SQL query(5) : M3

Récupération des résultats d'un select (3ème méthode)

$obj = mysql_fetch_object($result)

Retourne une ligne de résultat MySQL sous la forme d'un objet les attributs de l'objet portent le nom des champs de la table MySQL

•Exemple M3:

```
//affichage des champs id et comment de la table table_test
(object)
while ($tab = mysql_fetch_object($result)){
echo $tab->id . " : " . $tab->comment; echo "<br />";
}
mysql_close($id_connexion);
```

SQL query (injection SQL) 1

Le principal souci d'une bdd exposée sur le web provient des injections SQL:

L'injection SQL consiste en changer le but premier d'une requête SQL, grâce à des variables venant de l'utilisateur.

Il importe de se prémunir de ce type d'attaque i.e. de ne pas faire confiance aux données rentrées et de les filtrer….

Exemple : Imaginons une page php permettant de rechercher un utilisateur (enregistré en BD) qui se connecte par un formulaire où il indique son login et son mot de passe:

```
<form method="POST" action= "login.php" >
    <input type="text" name= "nom"><br>
    <input type="hidden" name= "passwd"><br>
     <input type="submit" value="Search">
</form>
```

La requête d'identification pourrait être quelque chose du style (login.php):

```
$query = "SELECT * FROM users WHERE user='{$_POST['nom']}'
AND password='{$_POST['passwd']}'";
```

Injection sql2

MAIS l'utilisateur a rentré :

$_POST['name'] = 'un_nom'; $_POST['password'] = "' OR ''='";

La requête exécutée est alors :

SELECT * FROM users WHERE user='un_nom' AND password='' OR ''=''

Voilà un beau trou de sécurité ouvert qui peut s'avérer destructeur… (détournement du select)

- une **technique répandue** : forcer l'analyseur SQL à ignorer le reste de la requête, en utilisant les symboles - - *ou* # pour mettre en commentaires la fin d' une requête ☹ *ou* /* *comment* */)

name : dupont' -- et passwd : nimportequoi donnera:

SELECT user_id WHERE user_name = 'dupont' -- ' AND user_password = '4e383a1918b432a9bb7702f086c56596e'

On exécute donc SELECT user_id WHERE user_name = 'dupont'

Le pirate peut alors se connecter sous l'utilisateur dupont avec n'importe quel mot de passe

Un exemple vrai d'injection sql

Supposons une table de **membres** ainsi définie :

```
CREATE TABLE membres (
    id int(10) NOT NULL auto_increment,
    login varchar(25),
    password varchar(25),
    nom varchar(30),
    email varchar(30),
    userlevel tinyint,
    PRIMARY KEY (id)
    )
```

Et qu'il existe un **script permettant de modifier son mot de passe,son nom,son mail** à partir d' un formulaire :

La requête SQL serait alors du type :

```
$sql = "UPDATE membres SET
    password='$pass',nom='$nom',email='$email' WHERE
    id='$id'";
```

Un exemple vrai d'injection sql

Un utilisateur malintentionné

- peut donner pour **$nom ',userlevel='3** :ce qui donnerait comme requête SQL :

UPDATE membres SET password='[PASS]',nom='',userlevel='3',email='[EMAIL]' WHERE id='[ID]'

//on modifie un champ de la table (si schéma connu)

- peut donner à $pass la valeur : [nouveaupass]' WHERE nom='Admin'#.

La requête deviendrait alors :

UPDATE membres SET password='[nouveaupass]' WHERE nom='Admin'#',nom='[NOM]',email='[EMAIL]' WHERE id='[ID]'

le SQL exécuté (donc sans les commentaires) sera :: (# :commentaire sql)

UPDATE membres SET password='[nouveaupass]' WHERE nom='Admin'

(on change le mot de passe de l'utilisateur admin…)

- peut donner $nom : **' OR 1=1"; drop table membres;#**
- c'est plus méchant car on va exécuter :

"UPDATE membres SET password='$pass',nom='' OR 1=1; drop table membres;# ,email='$email' WHERE id='$id'";

Des remèdes contre l'injection sql

La solution consiste à **traiter correctement les chaînes de caractères entrées par l'utilisateur**.

En PHP on peut utiliser pour cela la fonction **addslashes(),** qui transformera la chaîne ' -- en \' –

La requête deviendrait alors :

SELECT user_id WHERE user_name = 'dupont\' -- ' AND user_password = 'nimportequoi'

- **Ces attaques peuvent être évités de plusieurs façons :**

1. **Utiliser des requêtes préparées** / procédures stockées, à la place du SQL dynamique. Les données entrées par l'utilisateur sont alors transmises comme paramètres, sans risque d'injection.
2. **Vérifier** de manière précise et exhaustive l'ensemble des données venant de l'utilisateur. (is_numeric(), ou bien modifiez automatiquement le type avec la fonction settype(),)
3. **Utiliser des comptes utilisateurs SQL** à accès limité (en lecture-seule) quand cela est possible.
4. **Activer les protections internes** comme l'option *magic_quotes_gpc* en PHP(dans php_ini), qui échappe les guillemets simples, doubles, les / et NULL sur les GPC (GET,POST,COOKIES) ou utiliser addslashes (même effet mais à la demande)
5. **Se protéger du commentaire** sql : -- ou # ou /*

- **On dispose aussi de mysql_real_escape_string($chaine)**

Protège une chaîne pour la passer à mysql_query (la connexion à la base doit exister)

Sql : select sur plusieurs tables

personnes			
Id	Nom	Prénom	CP
number(10)	varchar(20)	varchar(20)	number(5)

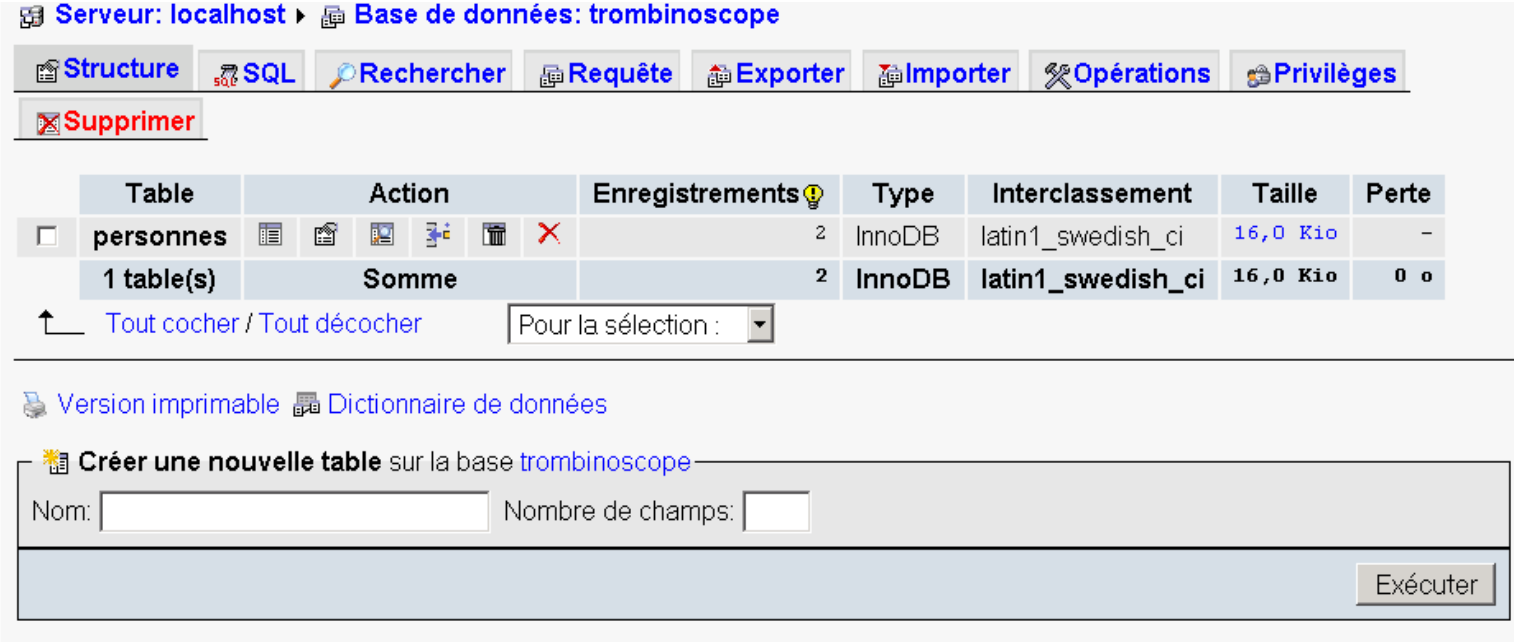
Code_postal	
nom_ville	CP
varchar(20)	Number(5)

- SELECT P.Id, P.nom,P.prenom,C.nom_ville FROM personnes P, code_postal C WHERE P.CP=C.cp

nous permet d'accèder au tableau

Id	Nom	Prénom	nom_ville

Exportation de base (ou de table)



- Permet d'obtenir le dump de la base (structures et/ou données)
- En divers formats(csv(;), sql,csv (tabulé),xml….)

Export CSV (; séparateur)

- 1;berger;jf;164364536535;M;berger.gif
- 2;dazy;jf;167345565456;M;dazy.gif

Exemple export SQL

```
CREATE TABLE `personnes` (
`id` int(10) NOT NULL auto_increment,
`nom` varchar(25) default NULL,
`prenom` varchar(25) default NULL,
`numero` varchar(30) default NULL,
`civilite` varchar(3) default NULL,
`nom_image` varchar(20) default NULL,
UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
--
-- Contenu de la table `personnes`
--
INSERT INTO `personnes` (`id`, `nom`, `prenom`, `numero`, `civilite`,
`nom_image`) VALUES
(1, 'berger', 'jf', '164364536535', 'M', 'berger.gif'),
(2, 'dazy', 'jf', '167345565456', 'M', 'dazy.gif');
```

Export en xml

```
<trombinoscope>
<!-- Table personnes -->
<personnes>
<id>1</id>
<nom>berger</nom>
<prenom>jf</prenom>
<numero>164364536535</numero>
<civilite>M</civilite>
<nom_image>berger.gif</nom_image>
</personnes>
<personnes>
<id>2</id>
<nom>dazy</nom>
<prenom>jf</prenom>
<numero>167345565456</numero>
<civilite>M</civilite>
<nom_image>dazy.gif</nom_image>
</personnes>
</trombinoscope>
```

Pour importer

Serveur: localhost ▸ Base de données: trombinoscope

Structure | SQL | Rechercher | Requête | Exporter | Importer | Opérations | Privilèges | Supprimer

Importer

Fichier à importer

Emplacement du fichier texte [ ] Parcourir... (Taille maximum: 2 048Kio)

Jeu de caractères du fichier: utf8

Ces modes de compression seront détectés automatiquement : aucune, gzip, zip

Importation partielle

☑ Permettre l'interruption de l'importation si la limite de temps est sur le point d'être atteinte. Ceci pourrait aider à importer des fichiers volumineux, au détriment du respect des transactions.

Nombre d'enregistrements (requêtes) à ignorer à partir du début 0

Format du fichier d'importation

◉ SQL

Options SQL

Mode de compatibilité SQL NONE

Exécuter

Ouvrir une nouvelle fenêtre phpMyAdmin

Exercice de synthèse (fin de cette séance et séance prochaine)

1. Créer une base nommée : trombinoscope
2. Créer une table nommée personnes avec les champs nom (majuscules), prénom (1ère lettre majuscule), civilité (M ou Mme), numéro ,nom_image (retaillée à 80 pixels de large en format jpeg)
3. Créer le programme permettant de peupler la table par un formulaire
4. (intégrant la création de la base et table si nécessaire)
5. Créer le programme permettant de supprimer un contact de la table par un formulaire
6. Créer le formulaire qui affiche les données d'un membre (formulaire avec un select dynamique) + script d'affichage
7. Version imprimable de l'annuaire (tri par ordre alphabétique)

1 : Création base trombinoscope et table personnes

1.1) Connexion

```
$link = mysql_connect("localhost", "mysql_user", "mysql_pw")
                    or die("Impossible de se connecter");
echo "Connexion réussie";
```

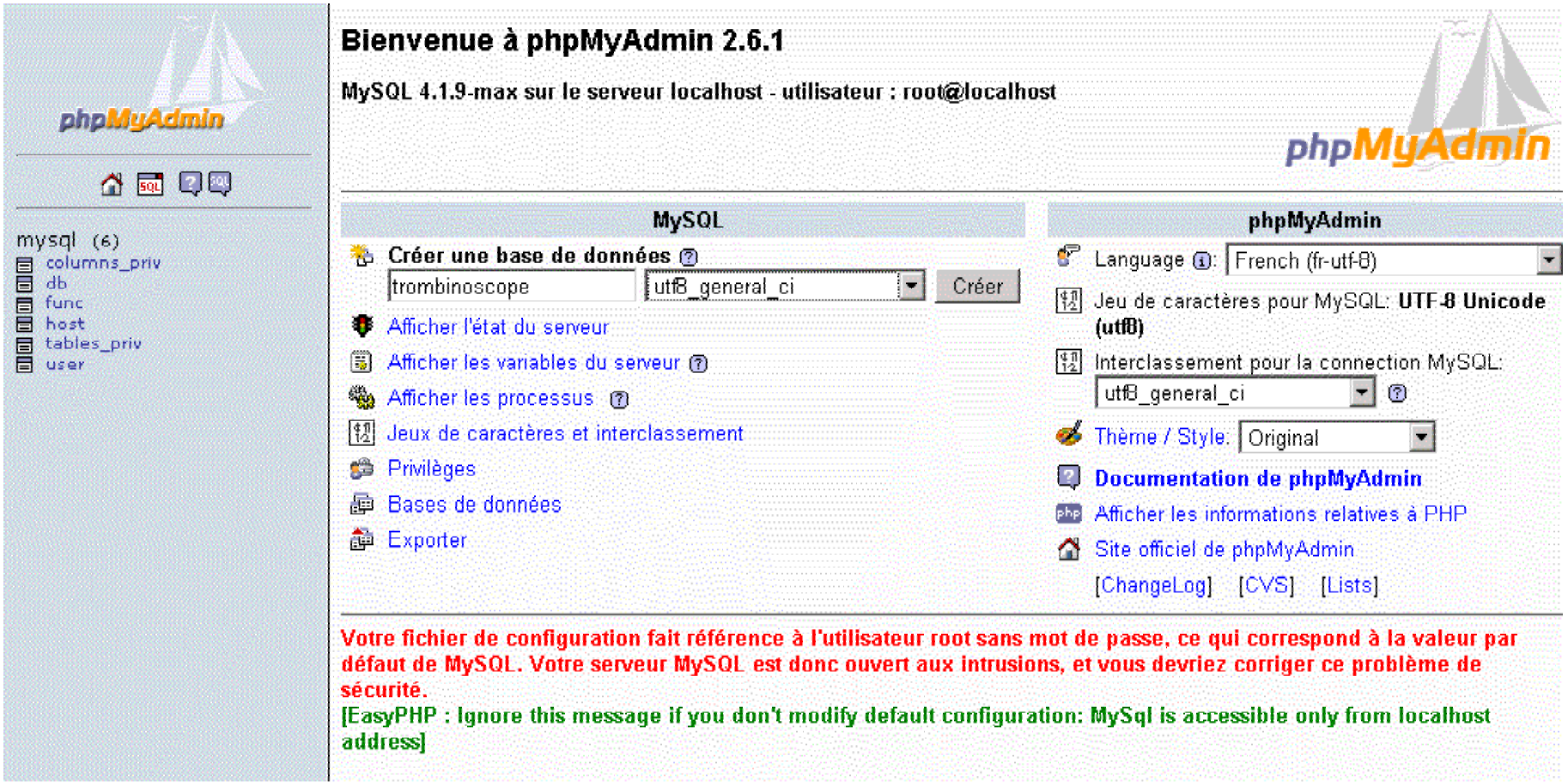
1.2) Base existante ou non (création de la base et de la table personnes si non), sélection de la base

```
If (!mysql_select_db("trombinoscope"))
{//création de la base et de la table si la base n'existe pas
$create_db='CREATE DATABASE trombinoscope DEFAULT CHARACTER SET latin1 ';
if (mysql_query($create_db, $link))
  {echo "Base de données créée correctement\n";
      mysql_select_db("trombinoscope");
  //création de la table personnes
    $create_personnes='CREATE TABLE personnes (
                                id int(10) NOT NULL auto_increment UNIQUE,
                                nom varchar(25),prenom varchar(25),
                                numero varchar(30),     civilite varchar(3),
                                nom_image varchar(20)) ';
  if (mysql_query($create_personnes, $link))
    {echo « table personnes créée correctement\n";}
  else
    {echo 'Erreur lors de la création de la table personnes : ';
     echo  mysql_error() . "\n";}
}
 else
  { echo 'Erreur lors de la création de la base de données : ';
   echo  mysql_error() . "\n";}
```

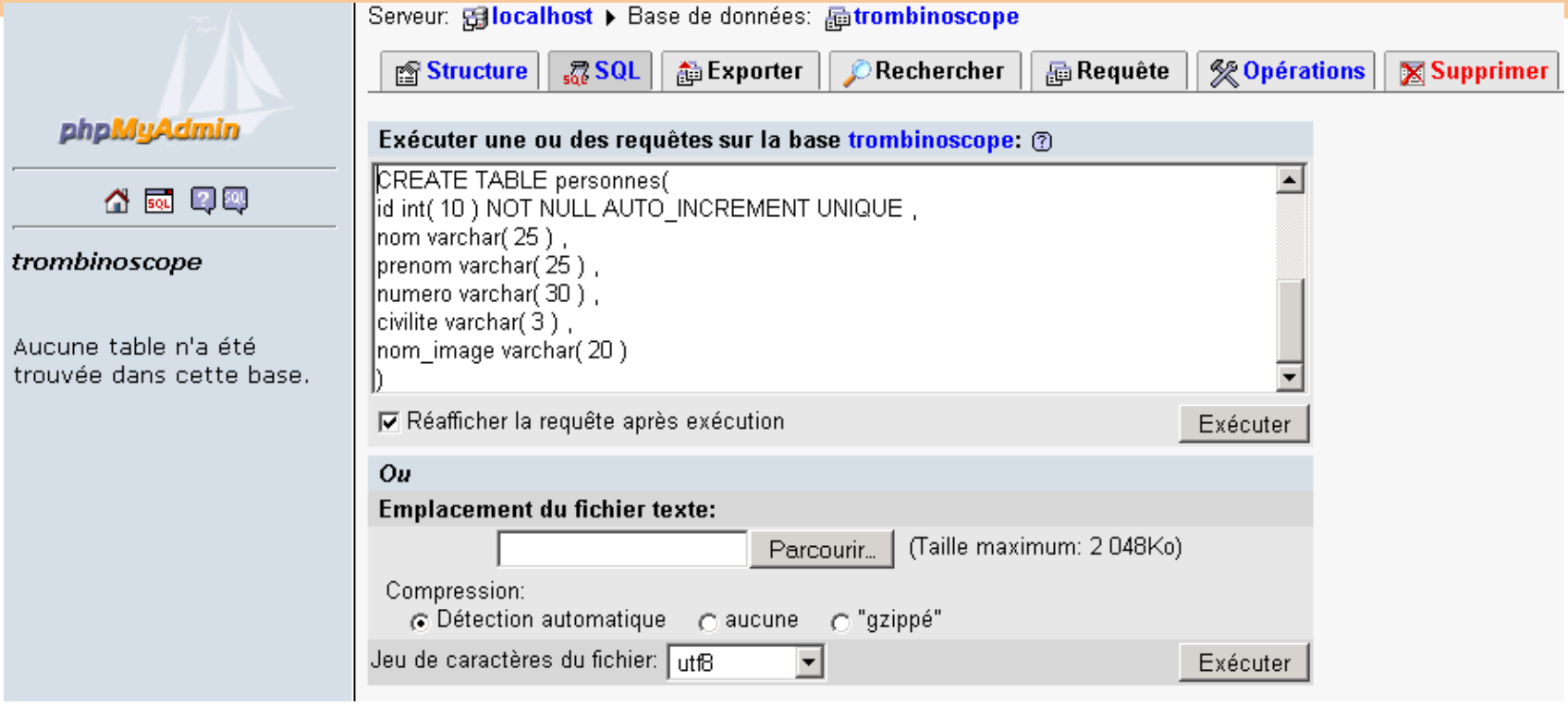
On peut aussi directement créer la table sous phpMyAdmin soit en rentrant les requêtes sql soit par l'outil

Création sous phpmyadmin base(1)

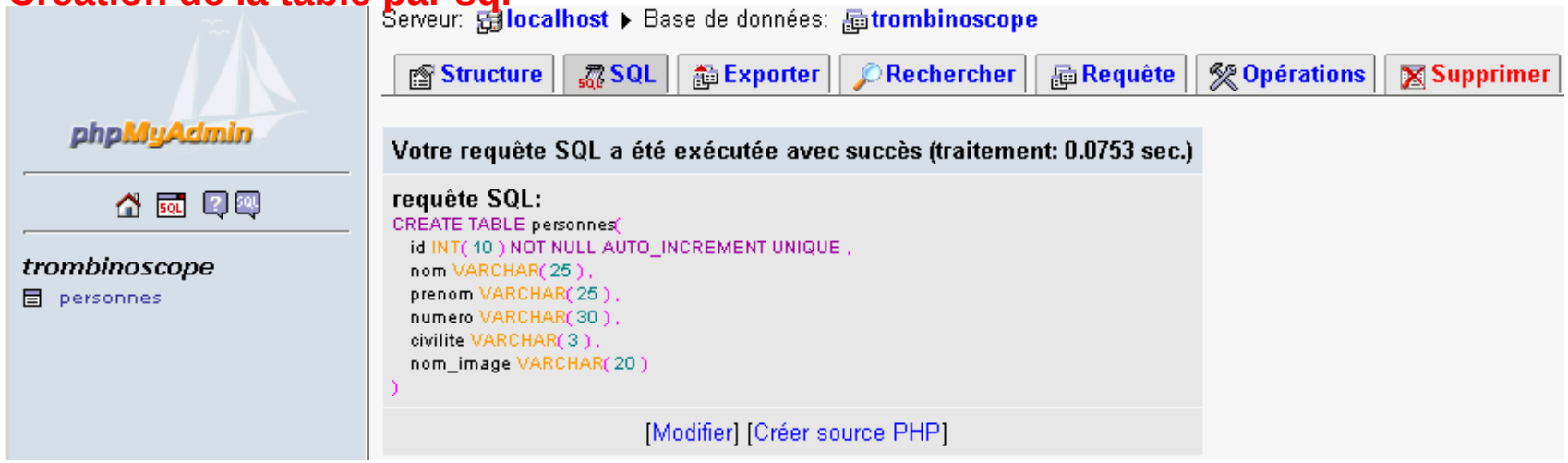
Création de la base trombinoscope



Création sous phpmyadmin table(2)



Création de la table par sql



Peuplement de la table (ajout_mysql.php)

```
<table summary="" border=1>
 <tr><td>Ajout d' un CONTACT</td>
      <td>
        <form action="ajout_mysql.php" method=POST
   ENCTYPE="multipart/form-data">
        <input type="hidden" name="MAX_FILE_SIZE" value="65536">
        <table width='100%'>
          <tr>
          <td width=25>M  <br>Mme</td>
         <td> <input type=checkbox value='M' name=civilite><br>
                <input type=checkbox value='Mme' name=civilite></td>
          <td width=100 align=right >Nom :</td>
          <td width=250 ><input type=text name=nom size=20></td>
         </tr>
```

… suite la page suivante…

Peuplement de la table (suite)

```
<tr>  <td colspan=3 align=right >Prénom : </td>
                <td><input type=text name=prenom size=20></td>      </tr>
          <tr>   <td colspan=3 align=right >Numéro : </td>
                <td><input type=text name=numero size=15></td>      </tr>
          <tr>   <td colspan=2 align=right >Image : </td>
                <td colspan=2><input type=file name="imagecontact"></td></tr>
          <tr><td colspan=4><i>l'image doit être au format gif, jpeg ou png et de taille
    inférieure à 64Koctets</td></tr>
          <tr><th colspan=4><input type=submit value="ajouter ce
    contact"></td></tr>
        </table>
    </form>
  </td>
 </tr>
</table>
```

Le formulaire d'ajout:

Trombinoscope avec mysql

Ajout d' un CONTACT	M ☐ Mme ☐ Nom : Prénom : Numéro : Image : Parcourir... *l'image doit être au format gif, jpeg ou png et de taille inférieure à 64Koctets* ajouter ce contact

Le formulaire d'ajout2

Le script appelé ***ajout_mysql.php*** doit :

- récupérer les éléments envoyés ($_POST et $_FILES)
- les vérifier (type de fichier, addslashes de champs, type d'images)
- les traiter (strtoupper, ucfirst, resize des images)
- se connecter à la base (création si nécessaire),
- enregistrer dans la table et copier l'image à l'emplacement voulu les éléments vérifiés et traiter
- traiter les problèmes rencontrés (informations…)

Bon travail....RDV le 30/3 sur ce sujet....