WEB DESIGNER
Approche développement

L'étudiant sera capable,
face à une structure informatique opérationnelle connectée à Internet, disposant des logiciels appropriés et de la documentation nécessaire, en utilisant le vocabulaire technique et l'orthographe adéquate, et en respectant les normes et standards en vigueur,

- d'identifier différents langages utilisés pour la programmation côté serveur ;
- d'installer les services nécessaires à l'exécution de scripts ;
- d'analyser un script serveur en termes de:
  - définition des variables,
  - structures conditionnelles et itératives,
  - fonctions et de procédures,
  - …
- d'exploiter un script serveur dans une page web ;
- d'utiliser, dans le langage choisi, les variables (de programmation, d'environnement, de session,…), les structures conditionnelles, les structures itératives, les tableaux, l'affichage dans une page web,… ;
- de transférer des données entre pages et scripts (méthodes GET et POST,…) ;
- de mettre en œuvre une application web dynamique ;
- d'interagir avec un système de gestion de bases de données (récupérer, ajouter, modifier, supprimer des enregistrements,…) ;
- de recourir à bon escient à la documentation disponible.
- d'analyser et de schématiser un problème donné et d'en dégager les besoins en termes de scripts serveurs et de bases de données ;
- de décrire et de caractériser une base de données ;
- de décrire et de caractériser les éléments essentiels d'un système de gestion de bases de données (SGBD) ;
- de schématiser une base de données à partir d'un problème pratique en justifiant les choix effectués ;

à l'aide d'un outil approprié :

- de créer une base de données par :
  - l'identification et la création de tables,
  - l'identification et la création d'index (clé primaire, clé étrangère,…),
  - l'identification des champs et la définition à bon escient du type de données,
  - l'identification et la création des relations entre les tables ;
- d'intervenir sur le contenu de la base de données par :
  - l'ajout, la modification et la suppression de données ;
- d'interroger une base de données ;
- utiliser un langage tel que SQL par :
  - des requêtes de sélection (simple, multiple, avec tri, avec filtre, avec jointure,…),
  - d'importer et d'exporter des données ;

Table des matières

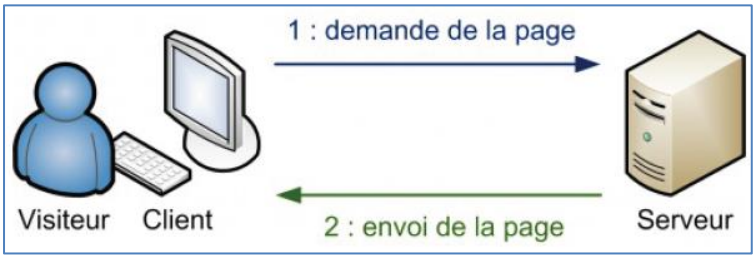
1. Introduction 5
1.1 Préparer son ordinateur 5
2. Premiers pas avec PHP 6
2.1 L'instruction echo 6
2.2 Inclure des portions de page 6
Exercice 1 : include 8
3. Les variables 8
3.1 Types de données 8
3.1.1 String (chaîne de caractères) 8
3.1.2 int (nombre entier) 8
3.1.3 Le type float (nombre décimal) 8
3.1.4 Le type bool (booléen) 8
3.1.5 Les tableaux 9
3.2 Afficher le contenu d'une variable 10
3.3 Les opérations de base : addition, soustraction… 10
Exercice 2 : les variables 11
4. Structures conditionnelles 11
4.1 if 11
4.2 Switch 11
4.3 Condition condensée 12
Exercice 3 : les conditions 12
5. Les répétitives 13
5.1 While 13
5.2 for 13
5.3 La boucle foreach 14
Exercice 4 : répétitive 14
6. Les fonctions 16
Exercice 5 : Les fonctions 18
7. Les formulaires : Transmettre des données de page en page 18
7.1 Les formulaires : la méthode POST 19
7.2 Passage de variables dans l'URL : la méthode GET 21
Exercice 6 : Les formulaires 21
8. Les variables session 24
9. Ecriture/lecture dans un fichier texte 25
Exercice 7 : les fichiers 27
Exercice 8 : utilisation de variable session 27
10. Gestion des bases de données – Théorie des bases de données relationnelles 28
1.1 Définitions 28

11. Administration avec l'outil phpMyAdmin 30
11.1 Présentation 30
11.2 Pourquoi utiliser la ligne de commande ? 30
12. Création d'une base de données 31
12.1 Identification et création de tables 31
12.2 Identification des champs et leurs types de données 32
12.2.1 Types des attributs 32
12.2.2 Entiers 32
12.2.3 Flottants 33
12.2.4 Chaînes 33
12.2.5 Dates et heures 34
12.2.6 Ensembles 34
12.3 Création des relations entre les tables 34
12.3.1 Clé primaire 34
12.3.2 Attribut non nul 34
12.3.3 Valeur par défaut 35
12.3.4 Attribut sans doublon 35
12.3.5 Index 35
12.3.6 Exercices – création de table 36
13. Gestion du contenu de la base de données 36
13.1 Ajout de données 36
13.2 Modification de données 37
13.3 Suppression de données 38
13.4 Exercices – insertion de données 38
14. Intégrité référentielle 39
14.1 Exercices – Analyse/ Intégrité référentielle 40
15. Mise à jour d'une base de données 42
15.1 Modification d'une table 42
15.1.1 Ajouter un attribut 42
15.1.2 Supprimer un attribut 42
15.1.3 Créer une clé primaire 43
15.1.4 Supprimer une clé primaire 43
15.1.5 Ajout d'une contrainte d'unicité 43
15.1.6 Changer la valeur par défaut d'un attribut 43
15.1.7 Changer la définition d'un attribut 44
15.1.8 Changer le nom d'une relation 44
15.1.9 Ajouter un index 44
15.1.10 Supprimer un index 44
15.2 Suppression d'une table 44

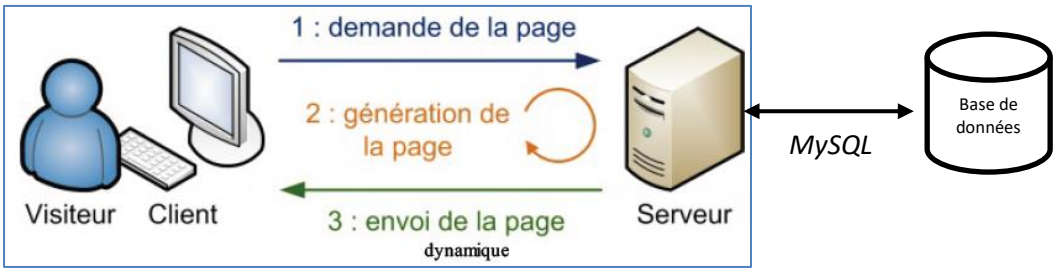
15.3 Exercices – mise à jour d'une base de données 45
16 Interrogation de base de données 46
16.1 Requêtes de sélection 46
16.1.1 Simple 46
16.1.2 Avec filtre 47
16.1.3 Avec tri 47
16.1.4 Exercices – Sélection simple/filtre/tri 48
16.1.5 Multiple 48
16.1.6 Avec regroupement 50
16.1.7 Champs calculés 52
16.1.8 Fonctions de MySQL 52
16.1.9 Les jointures internes 55
16.1.10 Les jointures externes 55
16.1.11 Exercices - jointures internes/externes/requêtes jointures/multiples/avec regroupement/champ calculée 58
16.2 Requêtes ensemblistes 60
16.2.1 Union 60
16.2.2 Intersection 60
16.2.3 Différence 61
16.2.4 Le produit cartésien 61
16.2.5 Exercices -Requêtes ensemblistes 62
16.2.6 Exercices récapitulatifs 62
17. Importation/Exportation des données 65
17.1 Réalisation d'un backup sous PHPMyAdmin 65
17.2 Restaurer une base de donnée sous PHPMyAdmin 66
17.2.1 Exemple d'importation sous OVH 67
17.3 Exporter une base de données MySQL 68
18. Interface avec PHP 69
18. 1 Choix de l'API 69
18. 2 PDO 69
18. 2.1 Activation - Php Windows 69
18.2.1 Connexion : 70
18.2.2 Gestion d'erreur 70
18.2.3 Méthodes : exec et query 71
11.2.4 Les requêtes préparées 72
18.3 Exercices 9 - PHP/ Base de données 74
19. Sources 75
20. Evaluation 76

1. Introduction

Les sites statiques : ce sont des sites réalisés uniquement à l'aide des langages HTML et CSS. Ils fonctionnent très bien mais leur contenu ne peut pas être mis à jour automatiquement : il faut que le propriétaire du site (le webmaster) modifie le code source pour y ajouter des nouveautés.



Les sites dynamiques : plus complexes, ils utilisent d'autres langages en plus de HTML et CSS, tels que PHP et MySQL. Le contenu de ces sites web est dit « dynamique » parce qu'il peut changer sans l'intervention du webmaster.



Le **PHP** est un langage exécuté par le serveur. Il permet de personnaliser la page en fonction du visiteur, de traiter ses messages, d'effectuer des calculs, etc. Il génère une page HTML.

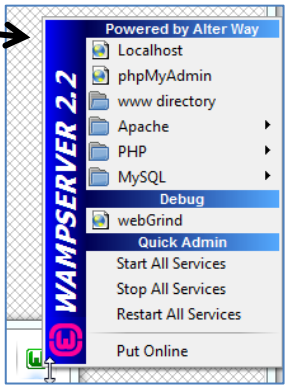
1.1 Préparer son ordinateur

Il existe des paquetages tout prêts pour Windows. WAMP Server a l'avantage d'être régulièrement mis à jour et disponible en français. http://www.wampserver.com/

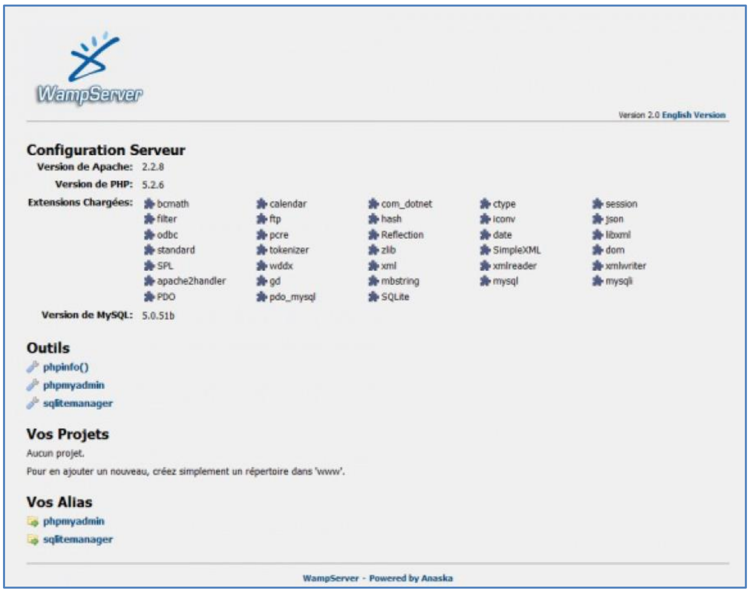
Wamp contient 3 choses :

- **Apache** : c'est ce qu'on appelle un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites web statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.
- **PHP** : c'est un plug-in pour Apache qui le rend capable de traiter des pages web dynamiques en PHP. En clair, en combinant Apache et PHP, notre ordinateur sera capable de lire des pages web en PHP.
- **MySQL** : c'est le logiciel de gestion de bases de données dont je vous ai parlé en introduction. Il permet d'enregistrer des données de manière organisée (comme la liste des membres de votre site). Nous n'en aurons pas besoin immédiatement, mais autant l'installer de suite.

Une fois le programme installé et démarré, vous pouvez alors lancer la page d'accueil de WAMP. Faites un clic gauche sur l'icône de WAMP puis cliquez sur *Localhost* :



La page d'accueil de wamp apparait alors :



Vos pages (reprises sous la rubrique « vos projets » se trouverons dans le dossier « C:\wamp\www »)
Il est conseillé d'utiliser un éditeur de texte qui colore le code source comme « Notepad++ » pour programmer convenablement en PHP.

2. Premiers pas avec PHP

- Les balises PHP commence par **<?php** et se termine par **?>** ; c'est à l'intérieur que l'on mettra du code PHP.
- Les commentaires longs s'écrivent en utilisant /* */

```
/*Ceci est un commentaire trèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèèè èèèèèèèèèèèèèèèèèèèèèèèèèèèèèèès long*/
```

- Les commentaires courts, sur une ligne s'écrivent //

```
//Ceci est un commentaire court
```

- Toutes les instructions PHP se terminent par un point-virgule.

2.1 L'instruction echo

Pour afficher du texte en PHP, on utilise l'instruction echo.

```
< ?php
        echo " Bonjour ";
?>
```

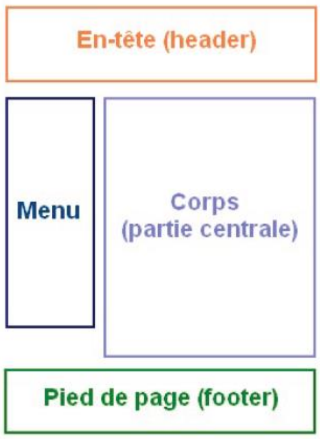
Enregistrer le fichier avec l'extension .php

<u>Remarque</u> :

Le mélange de balise html et de code se fait via l'opérateur de concaténation « **.** »

2.2 Inclure des portions de page

La plupart des sites web sont généralement découpés selon le schéma suivant :



```
<!doctype html>
<html lang=fr>
<head>
<meta charset="utf-8">
<title>Mon super site</title>
</head>
<body>
<!-- L'en-tête -->
<div id="en_tete">
</div>
<!-- Le menu -->
<div id="menu">
<div class="element_menu">
<h3>Titre menu</h3>
<ul>
<li><a href="page1.html">Lien</a></li>
<li><a href="page2.html">Lien</a></li>
<li><a href="page3.html">Lien</a></li>
</ul>
</div>
</div>
<!-- Le corps -->
<div id="corps">
<h1>Mon super site</h1>
<p>
Bienvenue sur mon super site !<br />
</p>
</div>
<!-- Le pied de page -->
<div id="pied_de_page">
<p>Copyright moi, tous droits réservés</p>
</div>
</body>
</html>
```

D'une page à l'autre, ce site contiendra à chaque fois le même code pour l'en-tête, le menu et le pied de page ! En effet, seul le contenu du corps change en temps normal.
Pour éviter de répéter le menu sur chacune des pages, nous utiliserons le principe de fonctionnement des **inclusions.**
Par exemple dans une page « menus.php » qui contiendrait uniquement :

```
<div id="menu">
<div class="element_menu">
<h3>Titre menu</h3>
<ul>
<li><a href="page1.html">Lien</a></li>
<li><a href="page2.html">Lien</a></li>
<li><a href="page3.html">Lien</a></li>
</ul>
</div>
</div>
```

Ensuite dans toutes les pages de votre site remplacez le menu par le code PHP suivant :

```
<?php include("menus.php"); ?>
```

Lorsque vous voudrez modifier votre menu, vous n'aurez qu'à modifier « menus.php » et l'ensemble des pages de votre site web sera automatiquement mis à jour !

Exercice 1 : include

- Créer trois pages html en utilisant la méthode vue ci-dessus pour le menu. (index.php, menu.php, pied.php page1.php,page2.php,…)

3. Les variables

Une **variable**, c'est une information stockée en mémoire **temporairement**. Elle n'a pas une grande durée de vie. En PHP, la variable (l'information) existe tant que la page est en cours de génération. Dès que la page PHP est générée, toutes les variables sont supprimées de la mémoire car elles ne servent plus à rien. Ce n'est donc pas un fichier qui reste stocké sur le disque dur mais une petite information temporaire présente en mémoire vive.

- Toutes les variables commencent par $
- Les variables ne sont pas typées par défaut

Chaque variable contiendra un **nom** : pour pouvoir la reconnaître (*Par exemple age_du_visiteur*) et une **valeur** : c'est l'information qu'elle contient, et qui peut changer. (*Par exemple : 17)*

3.1 Types de données

3.1.1 String (chaîne de caractères)

```
<?php
$nom_du_visiteur = "toto21";
$nom_du_visiteur = 'toto21';
?>
```

Remarque : si vous voulez insérer un guillemet simple alors que le texte est entouré de guillemets simples, il faut « l'échapper » en insérant un antislash devant. Il en va de même pour les guillemets doubles.

```
<?php
$variable = "Mon \"nom\" est toto21";
$variable = 'Je m\'appelle toto21';
?>
```

Ou encore, inverser les guillements :

```
<?php
$variable = 'Mon "nom" est Mateo21';
$variable = "Je m'appelle Mateo21";
?>
```

3.1.2 int (nombre entier)

```
<?php
$age_du_visiteur = 17;
?>
```

3.1.3 Le type float (nombre décimal)

Vous devez écrire votre nombre avec un point au lieu d'une virgule.

```
<?php
$poids = 57.3;
?>
```

3.1.4 Le type bool (booléen)

Pour dire si une variable vaut vrai ou faux, vous devez écrire le mot true ou false.

```
<?php
$je_suis_un_zero = true;
$je_suis_bon_en_php = false;
?>
```

3.1.5 Les tableaux

Il existe 2 sortes de tableaux :

- scalaires quand on travaille avec les indices ou des numéros
- associatif quand on travaille avec le label des champs.

3.1.5.1. Tableaux scalaires

```
$a[ ] ="titi" ;
$a[ ] ="toto" ;
```

Il n'est pas nécessaire de spécifier l'indice pour le remplissage du tableau, mais bien pour l'affichage :

```
echo $a[0] ;
```

Un tableau numéroté ou à indices est un tableau ou chaque case est identifiée par un numéro. Ce numéro est appelé clé. Attention, un tableau numéroté commence toujours à la case n°0

```
// Initialisation du tableau
$prenoms = array('Mathilde', 'Pierre', 'Amandine', 'Florian');

//Affichage des données
echo "Affichez le prénom qui a pour clé 0 <br>";
echo $prenoms[0] ."<br>";
// affichera Mathilde
echo $prenoms[2] ."<br>";
// affichera Amandine
```

3.1.5.2 Tableaux associatif

Les tableaux associatifs fonctionnent sur le même principe, sauf qu'au lieu de numéroter les cases, on va les étiqueter en leur donnant à chacune un nom différent.

```
$tab['nom']=  "Rémi" ;
echo $tab['nom'] ;
```

Vous remarquez qu'on écrit une flèche (=>) pour dire « associé à ».

```
// Initialisation du tableau
$ages = ['Mathilde' => 27, 'Pierre' => 29, 'Amandine' => 21];
//Affichage des données
echo "Affichez l'âge de Pierre<br >";
echo $ages['Pierre']."<br>";
```

Remarque :

Nous verrons plus loin que lors de l'envoi des données d'un formulaire par la méthode POST, la variable $_POST est un tableau associatif et chaque élément du tableau correspond au nom du champ du formulaire

```
echo $_POST ['nom'] ;
```

Cette instruction permettra de récupérer la valeur du tableau

3.1.5.3 Les tableaux multidimensionnels

Un tableau multidimensionnel est un tableau dont les valeurs peuvent elles-mêmes être des tableaux qui vont à nouveau pouvoir contenir d'autres tableaux et etc.

```
// Initialisation du tableau
$utilisateurs = [
   ['nom' => 'Tom', 'mail' => 'tom@ifosup.wavre.be'],
   ['nom' => 'Pierre', 'mail' => 'pierre@ifosup.wavre.be '],
   ['nom' => 'Amandine', 'mail' => 'amandine@ifosup.wavre.be']
];
```

```
//Affichage des données
echo "Affichez le nom de l'utilisateur qui a pour clé le nombre 2 <br>";
echo $utilisateurs[2]['nom']. '<br>';
// affichera Amandine
echo $utilisateurs[2]['mail']. '<br>';
// affichera amandine@ifosup.wavre.be
```

3.2 Afficher le contenu d'une variable

La fonction echo permet d'afficher le contenu d'une variable

```
<?php
$age_du_visiteur = 17;
echo $age_du_visiteur;
?>
```

```
<?php
$age_du_visiteur = 17;
echo "Le visiteur a ";
echo $age_du_visiteur;
echo " ans";
?>
```

Cette instruction peut aussi s'écrire en une seule ligne :

```
<?php
$age_du_visiteur = 17;
echo "Le visiteur a $age_du_visiteur ans";
?>
```

Ou encore :

```
< ?php
$age_du_visiteur = 17;
echo 'Le visiteur a ' . $age_du_visiteur . ' ans';
?>
```

3.3 Les opérations de base : addition, soustraction…

les quatre opérations de base + / * -

```
<?php
$nombre = 2 + 4; // $nombre prend la valeur 6
$nombre = 5 - 1; // $nombre prend la valeur 4
$nombre = 3 * 5; // $nombre prend la valeur 15
$nombre = 10 / 2; // $nombre prend la valeur 5

$nombre = 3 * 5 + 1; // $nombre prend la valeur 16
$nombre = (1 + 2) * 2; // $nombre prend la valeur 6
?>
```

Utilisation de parenthèse

```
<?php
$nombre = 10;
$resultat = ($nombre + 5) * $nombre; // $resultat prend la valeur
150
?>
```

Modulo % (reste de la division)

```
<?php
```

```
$nombre = 10 % 5; // $nombre prend la valeur 0 car la division
tombe juste
$nombre = 10 % 3; // $nombre prend la valeur 1 car il reste 1
?>
```

Exercice 2 : les variables

- 2.1 Réalisez un programme qui affiche le message: « Bonjour je suis un script PHP » – exe2.1.php
- 2.2 Réalisez un programme qui affichera en fonction de deux variables (jour et langue) le jour de la semaine dans la bonne langue. (Indice : il faut utiliser un tableau associatif à deux dimensions) – exe2.2.php

4. Structures conditionnelles

4.1 if

```
<?php
$age = 8;
if ($age <= 12)
{
echo "Salut gamin !";
}?
>
```

```
$age = 8;
if ($age <= 12) // SI l'âge est inférieur ou égal à 12
{
echo "Salut gamin ! Bienvenue sur mon site !<br />";
}else // Sinon
{
echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir entrer.!<br />";
}
?>
```

```
$age = 8;
if ($age <= 12)
{
    echo "Salut gamin !<br />";
}
elseif ($age<=18)           // Sinon Si…
{
    echo "salut l'ado!<br />";
}
else
{
    echo "salut le vieux!<br />";
}
?>
```

4.2 Switch

Voici un exemple avec un if assez lourd à la lecture…

```
$nombre=5 ;
if ($nombre<4){
   echo "le nombre est < que 4";
}elseif ($nombre <7){
```

```
        echo "le nombre est >3 et < 7";
       }elseif ($nombre =7){
              echo "le nombre est 7";
              }elseif ($nombre =8){
                     echo "le nombre est 8";
                     }elseif ($nombre =9){
                            echo "le nombre est 9";
                            }elseif ($nombre >9){
                                   echo "le nombre est >9"
                                   }
```

Celui-ci peut avantageusement être remplacé par un switch : (Selon que)

```
switch($nombre){
case 1:
case 2:
case 3:echo  "le nombre est < que 4";
          break;
case 4:
case 5:
case 6: echo "le nombre est >3 et < 7";
          break;
case 7: echo "le nombre est 7";
           break;
case 8: echo "le nombre est 8";
          break;
case 9: echo "le nombre est 9";
           break;
default: echo "le nombre est >9";
           break;
}
```

4.3 Condition condensée

```
 $age = 24;
if ($age >= 18)
{
$majeur = true;
}else
{
$majeur = false;
}
?>
```

Peut s'écrire en condensé :

```
<?php
$age = 24;
$majeur = ($age >= 18) ? true : false;
?>
```

La condition testée est $age >= 18. Si c'est vrai, alors la valeur indiquée après le point d'interrogation (ici true) sera affectée à la variable $majeur. Sinon, c'est la valeur qui suit le symbole « deux-points » qui sera affectée à $majeur.

Exercice 3 : les conditions

- 3.1 Ecrivez un programme (en utilisant un if) qui, en fonction d'une moyenne affichera le grade obtenu de l'étudiant. – exe3.1.Php

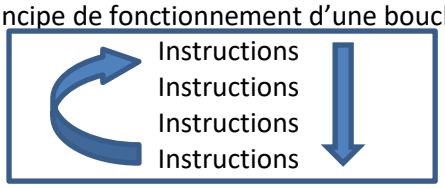
Exemple :

0≤moyenne <10 : refusé
10≤moyenne <14 : satisfaction
14≤moyenne <16 : distinction
16≤moyenne <18 : grande distinction
18≤moyenne <20 : la plus grande distinction

- 3.2 Même question en utilisant un switch – exe3.2.Php

5. Les répétitives

Principe de fonctionnement d'une boucle :



Concrètement, une boucle permet de répéter des instructions plusieurs fois.

5.1 While

```
<?php
while ($continuer_boucle == true)
{
// instructions à exécuter dans la boucle
}
?>
```

while peut se traduire par « tant que ». Ici, on demande à PHP : TANT QUE $continuer_boucle est vrai, exécuter ces instructions.

Par exemple, afficher les chiffres de 1 à 10 :

```
$chiffre= 1;
while ($chiffre <= 10)
{
echo $chiffre ;
$chiffre ++;            // $chiffre=$chiffre +1
}
?>
```

5.2 for

for et while donnent le même résultat et servent à la même chose : répéter des instructions en boucle.
L'une peut paraître plus adaptée que l'autre dans certains cas…

Le compteur sert à l'**initialisation**. C'est la valeur que l'on donne au départ à la variable

La **condition** : comme pour le while, tant que la condition est remplie, la boucle est réexécutée. Dès que la condition ne l'est plus, on en sort.

L'**incrémentation** vous permet d'ajouter 1 à la variable à chaque tour de boucle.

```
for (compteur; condition; modification du compteur) {
 liste d'instructions
}
```

Exemple :

```
<?php
for ($nombre_de_lignes = 1; $nombre_de_lignes <= 100; $nombre_de_lignes++)
{
echo 'Ceci est la ligne n°' . $nombre_de_lignes . '<br />';
}
?>
```

5.3 La boucle foreach

C'est une sorte de boucle for spécialisée dans les tableaux.

```
<?php
$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique','Benoît');

foreach($prenoms as $element)
{
echo $element . '<br />'; // affichera $prenoms[0], $prenoms[1] etc.
}
?>
```

L'avantage de foreach est qu'il permet aussi de parcourir les tableaux associatifs.

```
<?php
$coordonnees = array (
'prenom' => 'François',
'nom' => 'Dupont',
'adresse' => '3 Rue du Paradis',
'ville' => 'Marseille');

foreach($coordonnees as $element)
{
echo $element . '<br />';
}
?>
```

Toutefois, avec cet exemple, on ne récupère que la valeur. Or, on peut aussi récupérer la clé de l'élément. On doit dans ce cas écrire foreach comme ceci :

```
<?php foreach($coordonnees as $cle => $element) ?>
```

Remarque :

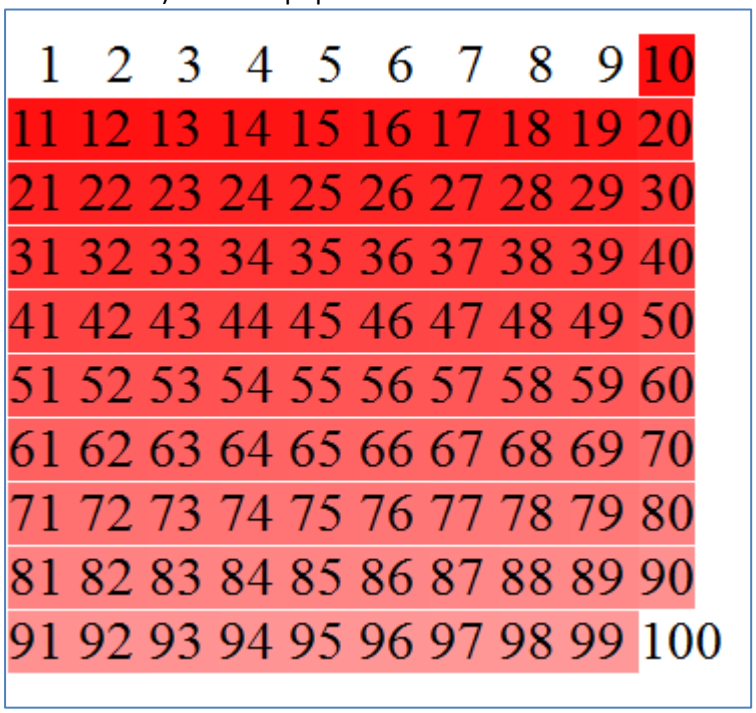
La fonction « print_r » permet d'afficher rapidement un tableau (c'est une sorte d'echo spécialisé dans les arrays).

Exercice 4 : répétitive

- 4.1 Affichez la table de multiplication par 8 (en utilisant une répétitive!) – exe4.1.php
- 4.2 Écrire un programme qui calcule la somme des entiers de 1 à 100 – exe4.2.php
- 4.3 Écrire un programme qui calcule le produit des entiers de 1 à 100 – exe4.3.php
- 4.4 Affichez les nombres suivant en HTML à l'aide d'une répétitive. – exe4.4.php

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

- 4.5 Reprenez l'exercice précédent et affectez une couleur de fond **différente** pour chaque chiffre (utilisez un <SPAN> en HTML) – exe4.5.php



- 4.6 Écrire le programme permettant d'afficher la table de multiplication suivante – exe4.6.php

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

- 4.7 Afficher les 50 premiers multiples de 5 – exe4.7.php

5 10 15 20 25 30 35.....250

- 4.8 Écrire un programme qui affiche les nombres de 20 à 1 de trois en trois. – exe4.8.php

Exemple :
20 19 18
17 16 15
14 13 12
...

(faire deux fois l'exercice : avec "for" et avec "while") – exe4.9.php

- 4.10 Affichez tous les nombres de 3 chiffres qui sont égaux à la somme des cubes de leurs chiffres (difficile !) – exe4.10.php
- 4.11 Affichez tous les nombres premiers plus petits que 1000(difficile !) – exe4.11.php
- 4.12 Affichez les nombres de 11 à 121 par pas de 11
- 4.13 Affichez un « carré » de « O » de 10 sur 10

O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O

O O O O O O O O O O
O O O O O O O O O O
O O O O O O O O O O

- 4.14 Affichez un triangle de « O »

O
OO
OOO
OOOO
OOOOO
OOOOOO
OOOOOOO
OOOOOOOO
OOOOOOOOO
OOOOOOOOOO

- 4.15 Affichez un triangle de « O » avec **aléatoirement**[1] des « X »

O
O O
O O O
O O X O
X X O O O
O O O O O O
O X O O O X O
O O O O X O X O
O O O O O O O O O
O O O O O X O O X O

6. Les fonctions

Comme les boucles, les fonctions permettent d'éviter d'avoir à répéter du code PHP que l'on utilise souvent.

Une fonction est une série d'instructions qui effectue des actions et qui retourne une valeur. En général, dès que vous avez besoin d'effectuer des opérations un peu longues dont vous aurez à nouveau besoin plus tard, il est conseillé de vérifier s'il n'existe pas déjà une fonction qui fait cela pour vous. Et si la fonction n'existe pas, vous avez la possibilité de la créer.

L'intérêt des fonctions réside dans le fait de clarifier le code et la possibilité de **réemploi** de ces fonctions.
Une fonction peut posséder plusieurs arguments qui seront séparés par des virgules. Une fonction renvoie généralement le résultat d'une demande : un nombre, une chaîne de caractère, ou une valeur booléenne (true, false). Elle doit être placée au début du script.

```
Function NomFonction(Argument1, Argument2)
        {
         liste d'instructions ;
        return $résultat ;
}
```

Les fonctions peuvent être assez complexes. Il est dès lors indispensable de les **commenter** en vue d'une utilisation ultérieure. (Détails pour les arguments, ce que la fonction retourne….)

Appel d'une fonction

```
Nom_De_La_Fonction () ;
```

Voici la liste des (nombreuses !) fonctions en php :
http://php.net/manual/fr/indexes.functions.php

[1] Utilisez la fonction rand() pour obtenir un nombre aléatoire

exemple de fonction php :

Traitement des chaînes de caractères

- strlen : longueur d'une chaîne
- str_replace : rechercher et remplacer
- strtolower : écrire en minuscules
- strtoupper : écrire en majuscules
- …

Récupérer la date

H = Heure
i = Minute
d = Jour
m = Mois
Y = Année

Pour afficher l'année :

```
<?php
$annee = date('Y');
echo $annee;
?>
```

```
<?php
// Enregistrons les informations de date dans des variables
$jour = date('d');
$mois = date('m');
$annee = date('Y');
$heure = date('H');
$minute = date('i');
// Maintenant on peut afficher ce qu'on a recueilli
echo 'Bonjour ! Nous sommes le ' . $jour . '/' . $mois . '/' . $annee . 'et il est ' . $heure. ' h ' . $minute;
?>
```

Création et utilisation d'une fonction

```
<?php
// Ci-dessous, la fonction qui calcule le volume du cône
function VolumeCone($rayon, $hauteur)
{
$volume = $rayon * $rayon * 3.14 * $hauteur * (1/3);        // calcul du volume
return $volume;        // indique la valeur à renvoyer, ici le volume
}
$volume = VolumeCone(3, 1);
echo 'Le volume d\'un cône de rayon 3 et de hauteur 1 est de ' .
$volume;
?>
```

Autres exemples de fonctions :

```
<?php
//déclaration de la fonction
function add($x,$y)
{
  $total=$x+$y;
  return $total;
}
//utilisation de la fonction
```

```
echo add(1,16);
?>
```

```
<?php
function div($nb1,$nb2)
{
 if ($nb2!=0)
   return $nb1/$nb2;
 else
   return "impossible";
}
//utilisation de la fonction
echo div(5,0)."</br>";
?>
```

```
<?php
function mult($a,$b)
{
 return $a*$b;
}
//utilisation de la fonction
echo mult(10,5)."</br>";
?>
```

Afin de ne pas surcharger le programme de base, il est souhaitable de regrouper les fonctions d'un programme dans un fichier séparer et le nommer lib.inc (=librairie)
A l'aide de l'instruction include ces fonctions peuvent être intégrées directement au programme

```
include ("lib.inc");
```

Exercice 5 : Les fonctions

- 5.1 Créez une fonction qui renvoie une chaîne de caractère en fonction de l'heure ("bonjour", "bon après-midi", "bonsoir", "bonne nuit", …) – exe5.1.php
- 5.2 Créer une fonction factorielle qui calcul la factorielle d'un nombre. – exe5.2.php

7. Les formulaires : Transmettre des données de page en page

[2]Un formulaire est constitué de champs et de contrôles : zones de texte, boutons radio, cases à cocher, listes déroulantes... Chacun de ces éléments se caractérise par un attribut « *name* » qui définira tout simplement le nom de la variable à récupérer en PHP avec la valeur rentrée par l'utilisateur.
Un formulaire HTML se définit avec les balises *<form>* et *</form>*. L'attribut "*action*" de cette balise permet de spécifier la page qui traitera les données fournies dans le formulaire par l'internaute ; l'attribut « *method* » la méthode de transmission (GET ou POST).

Ces deux méthodes peuvent être utilisées dans l'envoi de données via les formulaires. Nous préconisons l'emploi de la méthode POST car elle cache les informations transmises (qui transiteraient par l'URL dans le cas de la méthode GET). De plus, elle permet d'envoyer des données importantes en taille (la méthode GET se limite à 255 caractères) et assure la gestion de l'envoi de fichiers.

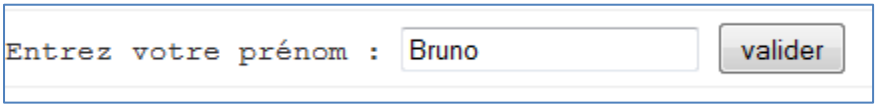
[2] Source : http://www.notre-planete.info/PHP/cours_8.php

7.1 Les formulaires : la méthode POST

Exemple simple de formulaire

```
<form action="resultat.php" method="post">
Entrez votre prénom : <input type="text" name="prenom" />
<input type="submit" value="valider" />
</form>
```

Ce qui donnera sur votre navigateur, si vous rentrez « Bruno » :



Sur cet exemple, l'utilisateur a la possibilité de spécifier une seule information : son prénom, cette donnée sera exploitable dans la page « resultat.php » sous la forme d'une variable **$_POST["prenom"]**[3] (composée de l'attribut « name » de la zone de texte à remplir) dont la valeur sera égale à la chaîne de caractères entrée.
Ainsi on aura par exemple dans la page « resultat.php » :

```
<?php
echo "Prénom tapé par l'utilisateur : ".$_POST['prenom'];
?>
```

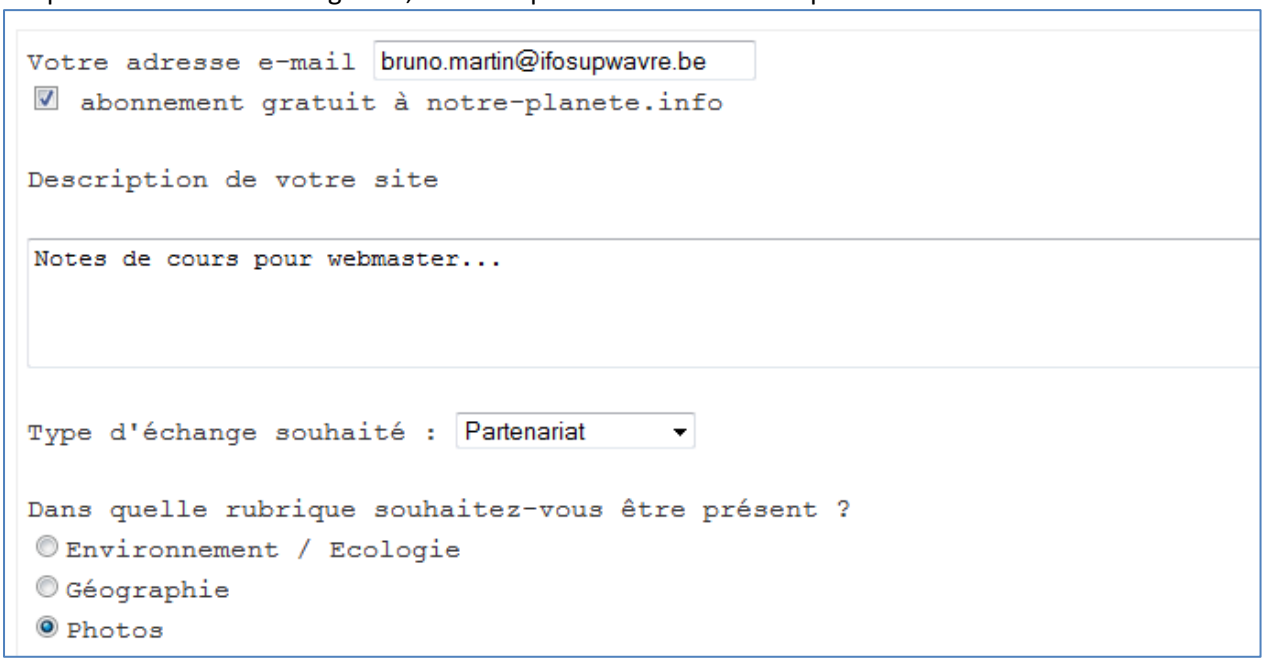
```
Prénom tapé par l'utilisateur : Bruno
```

Voici un exemple de traitement des variables qui émanent des différents contrôles et champs renseignés par un internaute sur un formulaire d'échange de liens.

```
<form method="POST" action="traitement.php">
Votre adresse e-mail<input type="text" name="mail" value="adresse e-mail" size="30" maxlength="55"><br/>
<input type="checkbox" NAME="mailing" value="oui" checked> abonnement gratuit à notre-planete.info<br/>
Description de votre site<br/>
<textarea name="description" rows="3" cols="60"></textarea><br/>
Type d'échange souhaité :<br/>
<select name="demande">
<option selected value="echange de liens">Echange de liens</option>
<option value="partenariat">Partenariat</option>
</select><br/>
Dans quelle rubrique souhaitez-vous être présent ?<br/>
<input type="radio" name="rubrique" value="environnement" />Environnement / Ecologie<br/>
<input type="radio" name="rubrique" value="geographie" />Géographie<br/>
<input type="radio" name="rubrique" value="photos"  />Photos<br/>
<input type="submit" value="valider" />
</form><br/>
```

[3] on peut écrire indistinctement : $_POST['prenom'], $_POST["prenom"] ou $_POST[prenom]

Ce qui donne sur votre navigateur, une fois qu'un demandeur a rempli le formulaire :



Nous remarquerons que l'appel à la page de traitement des données est « traitement.php » : on peut tout à fait appeler la même page en utilisant une variable globale PHP intitulée **$_SERVER['PHP_SELF']** qui signifie la page en cours.
Ainsi, l'affichage du formulaire et l'exploitation de ses résultats seront opérés sur la même page. Ceci implique que dans le code PHP, un test soit effectué pour savoir si le formulaire a été rempli ou non, par exemple, en testant si une des variables du formulaire a été déclarée avec la fonction **isset**().
Le code qui permet de récupérer les différentes valeurs saisies par l'utilisateur est le suivant :

```
<?php
$mail = $_POST["mail"];
echo "votre adresse e-mail est : $mail<br />";
$mailing = (isset($_POST["mailing"])? "oui" : "non");
echo "Avez-vous souhaité vous inscrire sur la lettre d'informations ? $mailing<br />";
$description = htmlentities($_POST["description"],ENT_QUOTES);
echo "et sa description est la suivante : <br />".nl2br($description)."<br />";
$rubrique =  $_POST["rubrique"];
echo "pour la rubrique ".$rubrique;
?>
```

Remarques :

- pour la case à cocher la variable prend la valeur true ou false si la case a été cochée ou non. Un simple test (écrit sous une forme raccourcie ici) attribue à la variable mailing la valeur "oui" ou "non". En fait, si la case a été cochée, la variable « name » correspondante est déclarée, sinon elle n'existe pas ;
- l'attribut « name » des boutons radios est le même car ils appartiennent à une même liste de choix. Le bouton radio sélectionné crée une variable nommée $_POST["rubrique"] qui est égale à la valeur notée dans l'attribut « value » s'il existe, sinon sa valeur sera égale à la chaîne de caractères spécifiée entre les balises <input> et </input> ;
- la liste d'options fonctionne de la même façon pour la valeur de la variable $_POST["demande"], l'attribut « name » étant déclaré dans la balise <select>
- nous utilisons la fonction **htmlentities**() qui transforme certains caractères du texte entré par l'utilisateur dans la zone de texte en d'autres qui ne permettront pas l'exécution de code HTML par une personne malintentionnée ;
- nous employons la fonction **nl2br**() qui permet de conserver les sauts de ligne effectués dans la zone de texte

votre adresse e-mail est : bruno.martin@ifosupwavre.be
Avez-vous souhaité vous inscrire sur la lettre d'informations ? oui
et sa description est la suivante :
Notes de cours pour webmaster.
pour la rubrique photos

Ces données pourraient ensuite être insérées dans une base de données pour y être conservées et exploitées par la suite.

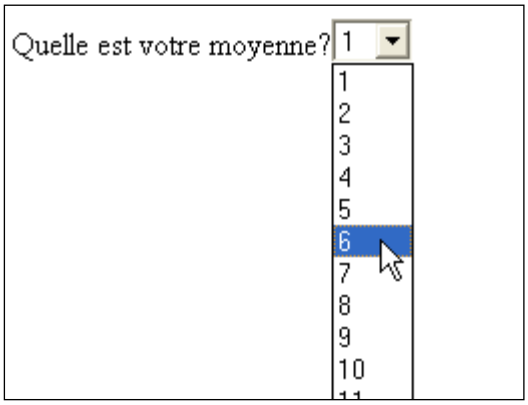
7.2 Passage de variables dans l'URL : la méthode GET

Dans ce cas, les variables et les valeurs qu'elles prennent sont déclarées directement dans l'URL c'est à dire via la balise de lien HTML <a href="URL">. Les variables sont ensuite exploitables sur la page cible en PHP.

La récupération des variables se fait via la syntaxe : **$variable = $_GET['variableurl']** au lieu de $_POST précédemment.

Exercice 6 : Les formulaires

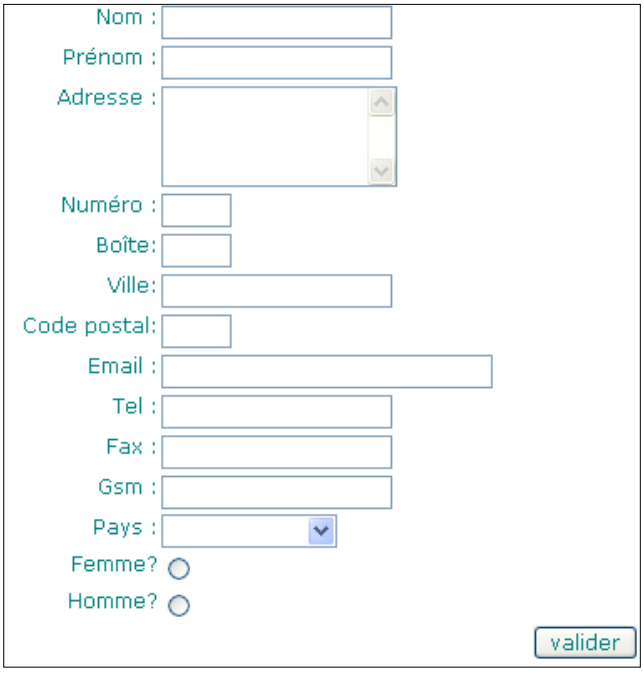
- 6.1 Ecrivez un programme qui, en fonction d'un formulaire, vérifiera le chiffre « posté » et affichera le grade obtenu de l'étudiant. **exe6.1.php – resultat6.1.php**



0≤moyenne <12 : refusé
12≤moyenne <14 : satisfaction
14≤moyenne <16 : distinction
16≤moyenne <18 : grande distinction
18≤moyenne <20 : la plus grande distinction

- 6.2 Réalisez un programme qui teste un login et un mot de passe transmis par un formulaire pour l'autorisation d'accès à une page. **exe6.2.php – resultat6.2.php**

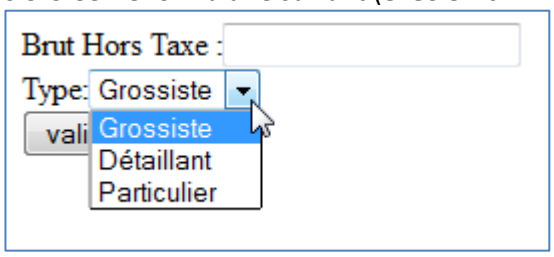
- 6.3 Affichez les résultats « postés » de votre formulaire sur une page php. **exe6.3.php – resultat6.3 .php**



Prévoyez au moins 5 pays dans la liste déroulante…

- 6.4 Envoyez un Email en php (formulaire + utilisation de la fonction mail) **exe6.4.php – resultat6.4.php**

- 6.5 Créez le formulaire suivant (**exe6.5.html – resultat6.5.php)**:



Celui sera renvoyé vers une page « remise.php », qui affichera la remise selon le tableau suivant

Si le type est un grossiste et qu'il a un brut hors taxes qui dépasse 10000 alors il a une remise de 10%
Sinon Si le type est un particulier et qu'il a un brut hors taxes qui dépasse 15000 alors il a une remise de 30%
Sinon Si le type est un détaillant et qu'il a un brut hors taxes qui dépasse 10000 alors il a une remise de 15%

- 6.6 **exe6.6.php – resultat6.6.php**

Prévoir un formulaire
Avec 2 champs :

- Âge (liste déroulante de 1 à 99)
- Jour de la semaine (liste déroulante avec tous les jours de la semaine)

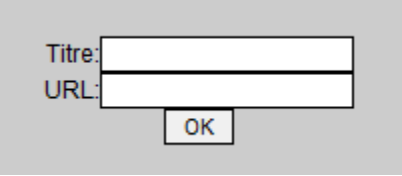
Ce formulaire redirigera vers une page (formulaire6.6.php) et affichera le prix d'entrée dans le parc d'attraction, sachant que :

- Les enfants de 3 à 11 ans paient 27€ en semaine et 29€ le we.
- Les personnes de 12 à 55 ans paient 31€ quelque soit le jour de la semaine
- Les autres ne paient pas l'entrée.

- 6.7 Scouts - Ecrivez un programme qui en fonction d'une variable informera l'enfant de sa catégorie **exe6.7.php – resultat6.7.php**

- Moins de 5 ans : trop petit !
- Baladins: de 5 à 8 ans
- Louveteaux : de 9 à 12 ans
- Scout : de 13 à 16 ans
- Pionniers : après 17 ans

- 6.8 **exe6.8.php – resultat6.8.php**. Réalisez le formulaire suivant :

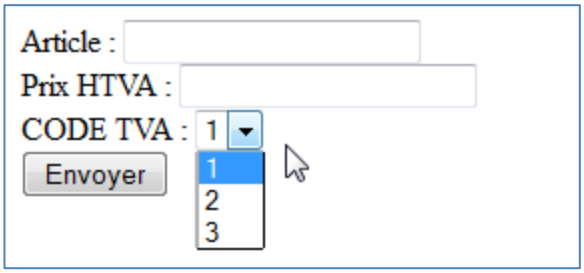


Celui-ci redirigera vers une page où vous devez récupérer les données et vérifier si le titre n'est pas vide et que l'URL commence bien par «http:// ».
Pour ce faire aidez-vous des deux fonctions ci-dessous :

- la fonction empty() permet de contrôler si un champ est vide ou non
- la fonction substr() permet de sélectionner une sous chaine dans une chaine (*par exemple, les 7 premiers caractères de l'URL…. Qui doivent être égale à « http:// »)*

http://be2.php.net/manual/fr/function.empty.php
http://be2.php.net/manual/fr/function.substr.php

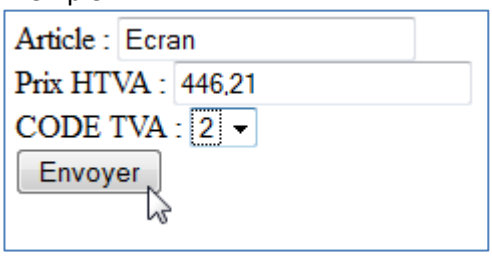
- 6.9 **exe6.9.php – resultat6.9.php**
  Prévoir un formulaire Avec 3 champs :
  - Nom de l'article
  - Prix HTV
  - Code TVA (prévoir pour ce champ une liste déroulante de 1 à 3)



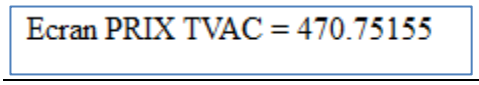
Ce formulaire redirigera vers une page et affichera en fonction du code TVA :
CODE TVA1 = 2,10%
CODE TVA2 = 5,50%
CODE TVA3 = 19,60%

Le nom de l'article + son prix TVAC

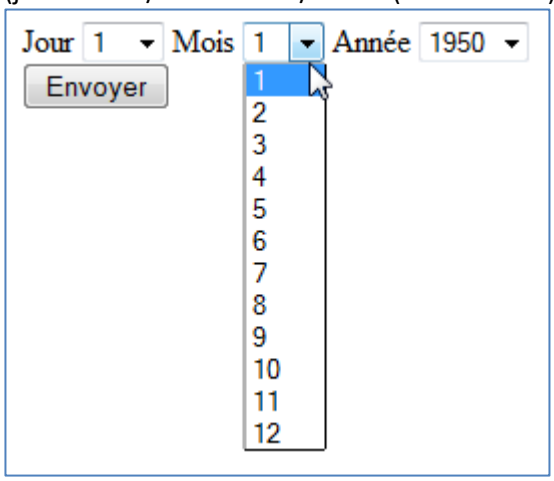
Exemple :



Affichera à l'écran :



- 6.10 **exe6.10.php – resultat6.10.php** Créez un formulaire qui permet de choisir dans des listes déroulantes (jour 1 à 31 /mois 1 à 12 /année (1950 à 2014) une date :



Affichez ensuite cette date à l'écran

8. Les variables session

Les sessions en PHP constituent le seul moyen sûr de suivre un utilisateur tout au long de sa visite du site. Grâce aux variables sessions qui se transmettent de page en page, pour chaque utilisateur au cours de sa visite.

Exemple :

Le formulaire suivant va ouvrir le fichier « formulaire.php »

```
<FORM NAME="FORM1" ACTION="formulaire.php" METHOD="POST">
<INPUT TYPE="TEXT" NAME="nom"></INPUT></br>
<INPUT TYPE="TEXT" NAME="prenom"></INPUT></br>
<INPUT TYPE="TEXT" NAME="adresse"></INPUT></br>
<INPUT TYPE="submit" NAME="submit" VALUE="Envoyer"></INPUT>
</FORM>
```

Dans le fichier « formulaire.php » on ouvre une session. Seule la valeur du champ 'nom' ='moi' donne la possibilité de voir le lien pour accéder à la page suivante et lecture du fichier « formulaire2.php »

```
<?php
session_start();
if ($_POST['nom']=="moi")
{
 $_SESSION['nom']=$_POST['nom'];
 $_SESSION['prenom']=$_POST['prenom'];
 $_SESSION['adresse']=$_POST['adresse'];
 echo "<A HREF='formulaire2.php'>Vers la page php2</A> ";
 }
else
{
 echo "Accès interdit";
}
?>
```

Transfert via le tableau associatif $_POST

Dans le fichier « formulaire2.php » le transfert de la variable se fait via le tableau $_SESSION

```
<?php
session_start();
if ($_SESSION['nom']=="moi")
{
```

```
 echo "<b>Bonjour</b></br>";
 echo $_SESSION['nom']."</br>";
 echo $_SESSION['prenom']."</br>";
 echo $_SESSION['adresse']."</br>";
 }
else
 echo "Acces interdit";
?>
```

9. Ecriture/lecture dans un fichier texte

fopen()

La fonction de base est la fonction fopen(). C'est elle qui permet d'ouvrir un fichier, que ce soit pour le lire, le créer, ou y écrire.

```
$pointeur = fopen(nomdufichier, mode);
```

Le mode indique le type d'opération qu'il sera possible d'effectuer sur le fichier après ouverture. Il s'agit d'une lettre (en réalité une chaîne de caractères) indiquant l'opération possible :

mode	Description
'r'	Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.
'r+'	Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
'w'	Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'w+'	Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'a'	Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
'a+'	Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
'x'	Crée et ouvre le fichier en lecture seule ; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, fopen va échouer, en retournant FALSE et en générant une erreur de niveau E_WARNING . Si le fichier n'existe pas, fopen tente de le créer. Ce mode est l'équivalent des options O_EXCL|O_CREAT pour l'appel système open(2) sous-jacent. Cette option est supportée à partir de PHP 4.3.2 et fonctionne uniquement avec des fichiers locaux.
'x+'	Crée et ouvre le fichier en lecture et écriture ; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, fopen va échouer, en retournant FALSE et en générant une erreur de niveau E_WARNING . Si le fichier n'existe pas, fopen tente de le créer. Ce mode est l'équivalent des options O_EXCL|O_CREAT pour l'appel système open(2) sous-jacent. Cette option est supportée à partir de PHP 4.3.2, et fonctionne uniquement avec des fichiers locaux.

fopen() retourne une ressource représentant le pointeur de fichier, ou **FALSE** si une erreur survient.
La fonction fopen () positionne toujours un pointeur dans le fichier. Une fois le fichier ouvert, on continue à travailler avec ce pointeur. C'est pour cette raison qu'au début, le statut du pointeur est conservé dans une variable.

Une fois que le fichier a été ouvert avec le mode désiré, il est possible de lire son contenu et d'y écrire des informations grâce aux fonctions : *fread()* et *fwrite()*

fread()

```
fread(pointeur ,length)
```

fread() lit jusqu'à length octets dans le fichier référencé par le pointeur . Si *length* est fourni, la lecture s'arrête lorsque length octets ont été lus, ou que l'on a atteint la fin du fichier.
fread() retourne la chaîne lue ou FALSE si une erreur survient.

fwrite()

```
fwrite(pointeur , TexteAécrireDansFichier , length )
```

fwrite() écrit le contenu de `TexteAécrireDansFichier` dans le fichier pointé. Si la longueur length est fournie, l'écriture s'arrêtera après length octets, ou à la fin de la chaîne (le premier des deux).
fwrite() retourne le nombre d'octets écrits ou FALSE en cas d'erreur.

fclose()

Un fichier ouvert avec la fonction *fopen()* doit être fermé, à la fin de son utilisation, par la fonction *fclose()* en lui passant en paramètre l'entier (le pointeur) retourné par la fonction *fopen()*

Exemple :
Soit le formulaire suivant

```
<FORM NAME="FORM1" ACTION="formulaire.php" METHOD="POST">
<INPUT TYPE="TEXT" NAME="nom"></INPUT></br>
<INPUT TYPE="TEXT" NAME="prenom"></INPUT></br>
<INPUT TYPE="submit" NAME="submit" VALUE="Envoyer"></INPUT>
</FORM>
```

fichier formulaire.php pour l'écriture

```
<?php
/* Récupération des données du formulaire dans une variable $tmp*/
$tmp=$_POST['nom'].";".$_POST['prenom']."\r\n";
//    \r\n ajout un passage à la ligne
/* Ouverture du fichier en mode "a" pour créer le fichier si celui-ci n'existe pas et le pointeur est placé en fin de fichier (mode ajout)*/
$fp=fopen("test.txt","a");
/* écriture dans le fichier test.txt du contenu de la variable $tmp */
fwrite($fp,$tmp);
//fermeture
fclose($fp);
?>
```

Si l'on souhaite lire les données du fichier texte créé précédemment

```
<?php
/* Ouverture du fichier test.txt uniquement en mode lecture "r" */
$fp=fopen("test.txt","r");
/*On lit l'entièreté du fichier (utilisation de filesize()) et on le place dans une variable $tmp*/
$tmp=fread($fp,filesize("test.txt"));
fclose($fp);
echo nl2br($tmp) ;//afficher le fichier txt avec les retours chariots.
?>
```

Le résultat est le suivant

```
Martin;Bruno
Martin;Tintin
Zieuvert;Bruno
```

filesize()

```
filesize(NomDuFichier)
```

filesize () renvoie la taille du fichier.

Isset()

```
bool  isset (var , mixed   var , ...  )
```

isset renvoie TRUE si la variable var est définie, FALSE sinon.

Exercice 7 : les fichiers

7.1 Réaliser un formulaire HTML contenant :

- 3 zone de texte : Nom, Prénom et Email
- ainsi que 2 boutons : Lire et Ecrire

Le bouton Lire permettra de lire le contenu du fichier texte et le bouton écrire permettra l'écriture du Nom et du Prénom dans le fichier texte

header()

```
header('Location: mapage.php');
```

la fonction header permet de rediriger le visiteur vers une autre page

explode()

```
array explode ( string $delimiter , string $string [, int $limit ] )
```

explode() retourne un tableau de chaînes, chacune d'elle étant une sous-chaîne du paramètre string extraite en utilisant le séparateur delimiter.

Exemple :

```
<?php
// Exemple 1
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
echo $pieces[1]; // piece2
?>
```

Exercice 8 : utilisation de variable session

8.1 Réaliser un formulaire qui vérifie un login et mot de passe dans un fichier texte « login.txt » qui a la structure suivante :

```
login;password
```

La page de réception du formulaire devra ensuite vérifier si le mot de passe et le login correspondent à ceux contenus dans le fichier texte, le cas échéant il redirigera vers une page « administration.php » qui créera une variable session de connexion (*par exemple connexion = true*) , sinon affichera à l'écran « Access denied ».
Vous aurez donc au final 4 fichiers :

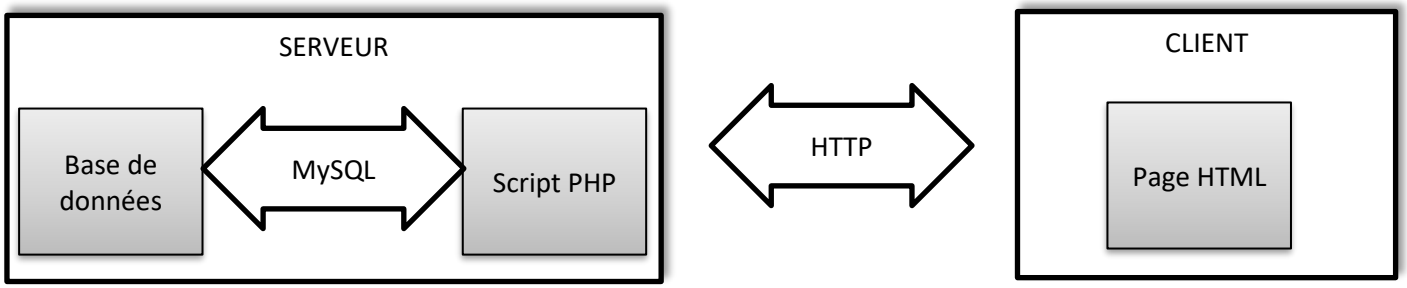
- Login.txt
- Formulaire.html
- Login.php (utilisation de la fonction ***explode*** pour « décortiquer » le fichier texte et utilisation de la fonction ***header*** pour la redirection vers la page « administration.php »
- Administration.php

10. Gestion des bases de données – Théorie des bases de données relationnelles

« En informatique, une base de données relationnelle est un stock d'informations décomposées et organisées dans des matrices appelées relations ou tables » ©Wikipedia

MySQL dérive directement de SQL (Structured Query Language) qui est un langage de requête vers les bases de données exploitant le modèle relationnel.

Le serveur de base de données MySQL est très souvent utilisé avec le langage de création de pages web dynamiques : PHP. Il sera discuté ici des commandes MySQL utilisables via PHP dans les conditions typiques d'utilisation dans le cadre de la gestion d'un site web.



1.1 Définitions

Domaine : ensemble des valeurs d'un attribut.

Relation : sous ensemble du produit cartésien d'une liste de domaines. C'est en fait un tableau à deux dimensions dont les colonnes correspondent aux domaines et dont les lignes contiennent des enregistrements (=tuples). On associe un nom à chaque colonne.
Une *relation* est une *table* comportant des *colonnes* (appelées aussi *attributs*) dont le *nom* et le *type* caractérisent le contenu qui sera inséré dans la table.
Imaginons que l'on veuille stocker dans notre base de données notre carnet d'adresses. On va donc créer la relation ***Personne*** qui aura pour attributs : *nom*, *prénom*, *adresse*, *téléphone*. Autrement dit, c'est une table nommée ***Personne*** possédant les colonnes : *nom*, *prénom*, *adresse*, *téléphone*.
Les *lignes* que contiendra cette table seront appelées *enregistrements ou tuples*.

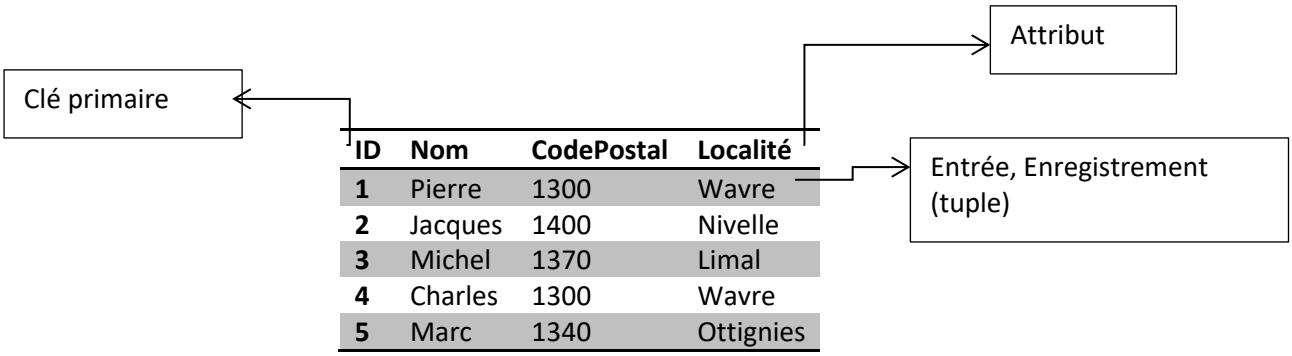
Personnes			
Nom	**Prenom**	**Adresse**	**telephone**
Dupond	Marc	8, Rue de l'octet	0123456789

L'algèbre relationnelle regroupe toutes les opérations possibles sur les relations.

Attribut : une colonne d'une relation, caractérisé par un nom.

Enregistrement ou Tuple : liste des valeurs d'une ligne d'une relation.
Une relation est un peu une classe (programmation orientée objet) qui ne possèderait que des attributs et donc chaque instance représenterait un enregistrement.

Clé primaire : ensemble minimal de colonnes qui permet d'identifier de manière **unique** chaque enregistrement.



ID	Nom	CodePostal	Localité
1	Pierre	1300	Wavre
2	Jacques	1400	Nivelle
3	Michel	1370	Limal
4	Charles	1300	Wavre
5	Marc	1340	Ottignies

Clé étrangère : référence dans la majorité des cas une clé primaire d'une autre table.

CP	Nom
1300	Wavre
1400	Nivelle
1370	Limal
1340	Ottignies

ID	Nom	CodePostal
1	Pierre	1300
2	Jacques	1400
3	Michel	1370
4	Charles	1340
5	Marc	1340

Une personne est domiciliée dans une localité.
Une localité est habitée par plusieurs personnes.
Il existe deux grands types de liens : *Un - Plusieurs* (comme le précédent) et *Plusieurs - Plusieurs*. La réalisation de ce dernier type de liens, un peu plus complexe, passe par l'utilisation d'une table intermédiaire dont la clé primaire est formée des clés étrangères des tables qu'elle relie.

Conclusion :

- Une base de données est un outil qui stocke vos données de manière organisée et vous permet de les retrouver facilement par la suite.
- On communique avec MySQL grâce au langage SQL. Ce langage est commun à tous les systèmes de gestion de base de données (avec quelques petites différences néanmoins pour certaines fonctionnalités plus avancées).
- PHP fait l'intermédiaire entre vous et MySQL.
- Une base de données contient plusieurs tables.
- Chaque table est un tableau où les colonnes sont appelées "champs" et les lignes "entrées".

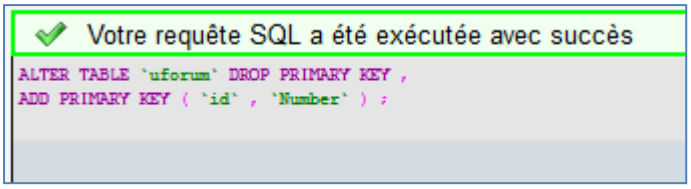
11. Administration avec l'outil phpMyAdmin

11.1 Présentation

L'outil phpMyAdmin est développé en PHP et offre une interface intuitive pour l'administration des bases de données du serveur.
Cet outil permet de :

- créer de nouvelles bases;
- créer/modifier/supprimer des tables;
- afficher/ajouter/modifier/supprimer des tuples dans des tables;
- effectuer des sauvegardes de la structure et/ou des données;
- effectuer n'importe quelle requête;
- gérer les privilèges des utilisateurs.

Toutes ces commandes sont disponible via cette interface graphique , toutefois, phpMyAdmin vous montrera toujours la commande SQL qui sera effectuée :



Il est possible d'utiliser MySql en mode "ligne de commande". Consultez les sites suivants pour plus d'information :

- https://www.sqlfacile.com/apprendre_bases_de_donnees/executer_requete_mysql_en_ligne_de_commande
- https://tecfa.unige.ch/guides/tie/html/mysql-intro/mysql-intro-7.html

11.2 Pourquoi utiliser la ligne de commande ?

- Les interfaces graphiques permettent de faire pas mal de choses, mais une fois que vous vous mettrez à faire des choses subtiles et compliquées, il faudra obligatoirement écrire vous-même vos requêtes ;
- Il est fort probable que vous désiriez utiliser MySQL en combinaison avec un autre langage de programmation. Or, dans du code PHP (ou Java, ou Python, etc.), on ne va pas écrire "Ouvre PhpMyAdmin et clique sur le bon bouton pour que je puisse insérer une donnée dans la base". On va devoir écrire en dur les requêtes. Il faut donc que vous sachiez comment faire.
- un des principal intérêt du SQL est la **portabilité**. Cela veut dire qu'un logiciel qui utilise une base de données peut fonctionner avec n'importe quelle base de données. Il suffira de lui indiquer avec quelle base de données il doit dialoguer. *(si pour une raison X, on doit changer la base, il suffit de modifier la relation entre le logiciel et la base de données)*

12. Création d'une base de données

- Chaque instruction SQL se termine par un point-virgule, mais dans PHPmyAdmin, lors de l'entrée directe des instructions SQL, elle est facultative;
- Les instructions SQL ne sont pas sensibles à la casse.

```
CREATE DATABASE NomBaseDeDonnees ;
```

12.1 Identification et création de tables

Création d'une table :

```
CREATE [TEMPORARY ] TABLE [IF NOT EXISTS] [NomBase.]Nom_de_la_table
        ( Colonne1 Type1 [NOT NULL | NULL] [DEFAULT valeur1] [COMMENT 'chaine1']
          Colonne2 Type2 [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaine2']
          Colonne3 Type3 [NOT NULL | NULL] [DEFAULT valeur3] [COMMENT 'chaine3']
          ...
        [CONSTRAINT nomContrainte1 typeContrainte1]
        ) ;
```

- TEMPORARY : créer une table qui n'existera que durant la session courante
- IF NOT EXISTS : éviter qu'une erreur se produise si la table existe déjà
- Nom_de_la_table : jusqu'à 64 caractères (sauf "/" "\" et ".")
- Colonnei typei : nom d'une colonne et son type (integer, char, date, …)
- DEFAULT : fixe la valeur par défaut
- NOT NULL | NULL : indique que la valeur de la cellule soit NULL ou pas
- COMMENT : permet de commenter une colonne
- NomContraintei typeContraintei : nom de la contrainte et son type (clé primaire, clé étrangère, etc.)

Exemple :

Un seul champ

Soit la création d'une table T_Copain, pourvue d'un seul champ texte NomClient de taille 50

```
CREATE TABLE T_Copain (NomClient VARCHAR(50));
```

2 champs

```
CREATE TABLE T_Copain
 (
 NomCopain varchar(30),
 Prenom    varchar(20)
 );
```

- Il est possible d'écrire sur plusieurs lignes pour rendre le code SQL plus clair
- Les champs sont séparés par des virgules

Avec une valeur par défaut, et l'interdiction de laisser un champ NULL

```
CREATE TABLE T_Copain
 (
 NomCopain VARCHAR (20) NOT NULL,
 Pays      VARCHAR (20) DEFAULT 'Belgique'
 );
```

Avec une clé primaire et un auto-incrément

L'exemple qui suit crée une table T_Copain, pourvue d'une clé primaire IDCopain, qui s'auto-incrémente. Utilisation de la fonction DATETIME dans le champ DateCreation (valeur par défaut dynamique)

```
CREATE TABLE T_Copain
 (
 IDCopain INT(11) auto_increment,
```

```
NomCopain VARCHAR(20),
DateCreation DATETIME,
PRIMARY KEY (IDCopain)
);
```

Cumul d'options de création de champs

```
CREATE TABLE T_Copain
(
IDCopain INT(11) auto_increment  PRIMARY KEY,
NomCopain VARCHAR(20),
DateCreation DATETIME
)
```

Remarque :

- Il est recommandé de préfixer par ID (ou pk_ pour primary key) les contraintes de clé primaires (IDRef (ou fk_ foreign key) les clés étrangères).
- Les mots SQL seront par convention tapés en majuscule, et les noms des champs/tables en minuscule sauf la première lettre.
- PRIMARY KEY = Unique + NOT NULL + Index

12.2 Identification des champs et leurs types de données

12.2.1 Types des attributs

Les propriétés de vos objets peuvent être de types très différents :

- Nombre entier signé ou non (température, quantité commandée, âge);
- Nombre à virgule (prix, taille);
- Chaîne de caractères (nom, adresse, article de presse);
- Date et heure (date de naissance, heure de parution);
- Énumération (une couleur parmi une liste prédéfinie);
- Ensemble (une ou des monnaies parmi une liste prédéfinie).

Il s'agit de choisir le plus adapté à vos besoins.
Ces types requièrent une plus ou moins grande quantité de données à stocker. Par exemple, ne pas choisir un LONGTEXT pour stocker un prénom mais plutôt un VACHAR(40) !

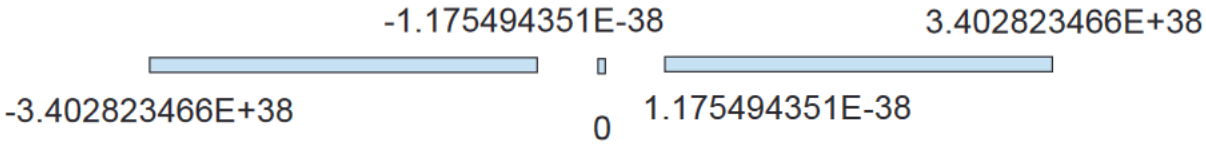
12.2.2 Entiers

nom	borne inférieure	borne supérieure
TINYINT	-128	127
TINYINT UNSIGNED	0	255
SMALLINT	-32768	32767
SMALLINT UNSIGNED	0	65535
MEDIUMINT	-8388608	8388607
MEDIUMINT UNSIGNED	0	16777215
INT*	-2147483648	2147483647
INT* UNSIGNED	0	4294967295
BIGINT	-9223372036854775808	9223372036854775807
BIGINT UNSIGNED	0	18446744073709551615

(*) : **INT** est un synonyme de **INTEGER**.
UNSIGNED permet d'avoir un type non signé.
ZEROFILL : remplissage des zéros non significatifs.

12.2.3 Flottants

Les flottants – dits aussi nombres réels – sont des nombres à virgule. Contrairement aux entiers, leur domaine n'est pas continu du fait de l'impossibilité de les représenter avec une précision absolue.
Exemple du type **FLOAT** :



nom	domaine négatif : borne inférieure borne supérieure	Domaine positif : borne inférieure borne supérieure
FLOAT	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
DOUBLE*	-.7976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

(*) : **DOUBLE** est un synonyme de **REAL**.

12.2.4 Chaînes

nom	longueur
CHAR(M)	Chaîne de taille fixée à M, où 1<M<255, complétée avec des espaces si nécessaire.
CHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
VARCHAR(M)	Chaîne de taille variable, de taille maximum M, où 1<M<255, complété avec des espaces si nécessaire.
VARCHAR(M) BINARY	Idem, mais insensible à la casse lors des tris et recherches.
TINYTEXT	Longueur maximale de 255 caractères.
TEXT	Longueur maximale de 65535 caractères.
MEDIUMTEXT	Longueur maximale de 16777215 caractères.
LONGTEXT	Longueur maximale de 4294967295 caractères.
DECIMAL(M,D)	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupent un caractère.

Remarques :
"Les types `CHAR` et `VARCHAR` sont similaires, mais diffèrent dans la manière dont ils sont stockés et récupérés.
La longueur d'une colonne `CHAR` est fixée à la longueur que vous avez défini lors de la création de la table. La longueur peut être n'importe quelle valeur entre 1 et 255. (Dans la version 3.23 de MySQL, la longueur est comprise entre 0 et 255.) Quand une valeur `CHAR` est enregistrée, elle est complétée à droite avec des espaces jusqu'à atteindre la valeur fixée. Quand une valeur de `CHAR` est lue, les espaces en trop sont retirés.
Les valeurs contenues dans les colonnes de type `VARCHAR` sont de tailles variables. Vous pouvez déclarer une colonne `VARCHAR` pour que sa taille soit comprise entre 1 et 255, exactement comme pour les colonnes `CHAR`. Par contre, contrairement à `CHAR`, les valeurs de `VARCHAR` sont stockées en utilisant autant de caractères que nécessaire, plus un octet pour mémoriser la longueur. Les valeurs ne sont pas complétées. Au contraire, les espaces finaux sont supprimés avant stockage (ce qui ne fait pas partie des spécifications ANSI SQL).
Si vous assignez une chaîne de caractères qui dépasse la capacité de la colonne `CHAR` ou `VARCHAR`, celle-ci est tronquée jusqu'à la taille maximale du champ."

NUMERIC est un synonyme de **DECIMAL**.
Les types TINYTEXT, TEXT, MEDIUMTEXT et LONGTEXT peuvent être judicieusement remplacés respectivement par TINYBLOB, BLOB, MEDIUMBLOB et LONGBLOB.
Ils ne diffèrent que par la sensibilité à la casse qui caractérise la famille des BLOB. Alors que la famille des TEXT est insensible à la casse lors des tris et recherches.
Les BLOB peuvent être utilisés pour stocker des données binaires.
Les VARCHAR, TEXT et BLOB sont de taille variable. Alors que les CHAR et DECIMAL sont de taille fixe.

12.2.5 Dates et heures

nom	description
DATE	Date au format anglophone AAAA-MM-JJ.
DATETIME	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(M)	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de TIMESTAMP.
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

nom	description
TIMESTAMP(2)	AA
TIMESTAMP(4)	AAMM
TIMESTAMP(6)	AAMMJJ
TIMESTAMP(8)	AAAAMMJJ
TIMESTAMP(10)	AAMMJJHHMM
TIMESTAMP(12)	AAMMJJHHMMSS
TIMESTAMP(14)	AAAAMMJJHHMMSS

En cas d'insertion d'un enregistrement en laissant vide un attribut de type TIMESTAMP, celui-ci prendra automatiquement la date et heure de l'insertion. Contrairement à Unix (où le timestamp est le nombre de secondes écoulées depuis le 1er janvier 1970), en MySQL, il est une chaîne de format comme indiqué ci-contre.

12.2.6 Ensembles

nom	description
ENUM('valeur','valeur2',...)	Une énumération ENUM est une chaîne dont la valeur est choisie parmi une liste de valeurs autorisées lors de la création de la table. (65535 valeurs max.)
SET('valeur','valeur2',...)	Un SET est une chaîne qui peut avoir zéro ou plusieurs valeurs, chacune doit être choisie dans une liste de valeurs définies lors de la création de la table. (64 valeurs max.)

12.3 Création des relations entre les tables

12.3.1 Clé primaire

Toute table doit posséder un champ qui joue le rôle de clé primaire. La clé primaire permet d'identifier de manière unique une entrée dans la table. Par exemple, chaque article de votre site Internet doit pouvoir être identifié de manière unique. Le moyen le plus simple pour cela est de lui donner un numéro unique, dans un champ nommé "id". **Il ne peut pas y avoir deux articles avec le même id** – si deux articles ont le même numéro, on ne pourra pas les différencier !

Exemples de clé primaire : Numéro de registre national, numéro ISBN, ADN, etc…

12.3.2 Attribut non nul

Considérons que l'on souhaite que certains attributs aient obligatoirement une valeur. On utilisera l'option **NOT NULL**.

Dans ce cas, si malgré tout, aucune valeur n'est fournie, la valeur par défaut – si elle est déclarée à la création de la relation – sera automatiquement affectée à cet attribut dans l'enregistrement.

Si aucune valeur par défaut n'est déclarée :

- la chaîne vide **" "** sera affectée à l'attribut s'il est de type chaîne de caractères;
- la valeur zéro **0** s'il est de type nombre;
- la date nulle 0000-00-00 et/ou l'heure nulle 00:00:00 s'il est de type date, heure ou date et heure.

Exemple :

```
adresse TINYTEXT NOT NULL
```

Au contraire, on utilisera l'option **NULL** si on autorise l'absence de valeur.

12.3.3 Valeur par défaut

Pour donner une valeur par défaut à un attribut, on utilise l'option **DEFAULT**.
Lors de l'ajout d'un enregistrement cette valeur sera affectée à l'attribut si aucune valeur n'est donnée.
Exemple :

'telephone' DECIMAL(10,0) DEFAULT '0123456789'

Les attributs de type chaîne de caractères de la famille TEXT et BLOB ne peuvent pas avoir de valeur par défaut.

12.3.4 Attribut sans doublon

Pour interdire l'apparition de doublon pour un attribut, on utilise l'option **UNIQUE**.
Syntaxe :

UNIQUE [nomdelacontrainte**](**liste des attributs**)**

Exemple, pour interdire tout doublon de l'attribut *nom* :

UNIQUE(*nom*)

Pour interdire les doublons sur l'attribut *nom* mais les interdire aussi sur *'prénom'*, tout en les laissant indépendants :

UNIQUE(*nom*)
UNIQUE(*prenom*)

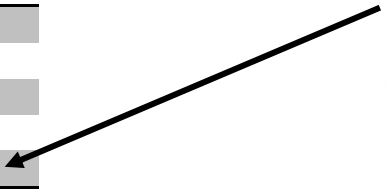
nom	prenom
Dupond	**Marc**
Dupont	Pierre
Martin	**Marc**

Enregistrement interdit car 'Marc' est un doublon dans la colonne 'prenom'

Pour interdire tout doublon à un ensemble d'attributs (tuple), on passe en paramètre à **UNIQUE** la liste des attributs concernés.
Pour interdire tout doublon du couple (*nom*, *prenom*) :

UNIQUE(nom,prenom**)**

nom	prenom
Dupond	Marc
Dupont	Pierre
Martin	**Marc**
Martin	Pierre
Martin	**Marc**



Enregistrement interdit car le couple ('Martin', 'Marc') est un doublon du couple (nom,prenom)

12.3.5 Index

Lors de la recherche d'informations dans une relation, MySQL parcourt la table correspondante dans n'importe quel ordre. Dans le cas d'un grand nombre de lignes, cette recherche est très très longue du fait du parcours de TOUTE la table.
Pour y remédier, une optimisation possible et FORTEMENT recommandée, est d'utiliser des index.
La création d'un index associé à un attribut ou à un ensemble ordonné d'attributs va créer une liste ordonnée des valeurs de ces attributs et de l'adresse de la ligne associée. C'est sur les valeurs de cette liste que se feront les recherches et les tris. Les algorithmes de recherche et de tri sur des ensembles ordonnés sont énormément plus rapides !
Ainsi, d'une recherche à coût prohibitif, on passe à une recherche sur un ensemble déjà trié. On gagne donc énormément en temps d'accès aux informations. Bien que cela ralentisse les mises à jour (insertion, suppression, modification de clé).
On choisira de créer des index sur les attributs qui seront les plus sollicités par les recherches ou utilisés comme critère de jointure. Par contre, on épargnera les attributs qui contiennent peu de valeurs différentes les unes des autres et ceux dont les valeurs sont très fréquemment modifiées.

Syntaxe :

INDEX index (liste des attributs**)**

Exemple, pour créer un index sur les 3 premiers caractères seulement de l'attribut *nom* :

```
INDEX idx_nom (nom(3))
```

Exemple, pour créer un index sur le couple (*nom*,prenom) :

```
INDEX idx_nom_prenom (nom,prenom)
```

- Un index peut porter sur 15 colonnes maximum.
- Une table peut posséder au maximum 16 index.
- Un index peut avoir une taille d'au maximum 256 octets et ne doit porter que sur des attributs NOT NULL.

12.3.6 Exercices – création de table

- Créez un système d' « article» pour votre site. Les informations relatives à un article sont les suivantes : titre, texte, date de parution, auteur, rubrique (soit "économie", "sports", "international", "politique" ou "culture")

- Créez une table "T_contact" qui contiendrait toutes les informations relatives aux personnes de votre carnet d'adresses. Quels champs seraient-ils utiles d'indexer ?

13. Gestion du contenu de la base de données

13.1 Ajout de données

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] [nomBase.] { nomTable | nomVue } [(nomColonne,...)]
VALUES ({expression | DEFAULT},...),(...),...
[ON DUPLICATE KEY UPDATE nomColonne = expression,...]
```

- `DELAYED` indique que l'insertion est différée (si la table est modifiée par ailleurs, le serveur attend qu'elle se libère pour y insérer périodiquement de nouveaux enregistrements si elle redevient active entre-temps).
- `LOW_PRIORITY` indique que l'insertion est différée à la libération complète de la table (option à ne pas utiliser sur des tables `MyISAM`).
- `HIGH_PRIORITY` annule l'option *low priority* du serveur.
- `IGNORE` indique que les éventuelles erreurs déclenchées suite à l'insertion seront considérées en tant que *warning*s.
- `ON DUPLICATE KEY UPDATE` permet de mettre à jour l'enregistrement présent dans la table, qui a déclenché l'erreur de doublon (dans le cas d'un index `UNIQUE` ou d'une clé primaire). Dans ce cas le nouvel enregistrement n'est pas inséré, seul l'ancien est mis à jour.

Ajouter un enregistrement à une relation revient à ajouter une ligne à la table. Pour cela, pour chacun des attributs, il faudra en préciser la valeur. Si certaines valeurs sont omises, alors les valeurs par défaut définies lors de la création de la relation seront utilisées. Si on ne dispose pas non plus de ces valeurs par défaut, alors MySQL mettra 0 pour un nombre, '''' pour une chaîne, 0000-00-00 pour une date, 00:00:00 pour une heure, 00000000000000 pour un timestamp (si la contrainte NOT NULL est présente). Dans le cas où l'attribut porte la contrainte NULL (par défaut) alors la valeur par défaut de l'attribut – quel soit son type – sera la suivante : NULL.

Syntaxe d'une "insertion étendue" :

```
INSERT INTO Table(liste des attributs) VALUES(liste des valeurs)
```

Exemple :

```
INSERT INTO Personnes(nom,prénom) VALUES('Martin','Paolo')
```

REPLACE est un synonyme de INSERT, mais sans doublon. (Respect des contraintes d'unicité (UNIQUE, PRIMARY KEY).

Une syntaxe plus courte mais plus ambiguë permet d'insérer un enregistrement dans une table. Elle consiste à omettre la liste des noms d'attribut à la suite du nom de la relation. Cela impose que la liste des valeurs suivant le

mot clé VALUES soit exactement celle définie dans la table et qu'elle soit dans l'ordre défini dans la définition de la table ; sinon des erreurs se produiront.

Syntaxe d'une "insertion standard" :

INSERT INTO *table* VALUES(*liste exhaustive et ordonnée des valeurs***)**

Exemple :

```
CREATE TABLE Ballon (
        taille INT NOT NULL,
        couleur VARCHAR(40)
)
```

INSERT INTO *Ballon* VALUES(20, 'rouge') →ok

INSERT INTO *Ballon* VALUES('rouge', 20) →faux

INSERT INTO *Ballon* VALUES('rouge') →faux

Syntaxe d'une "insertion complète ":

INSERT INTO relation VALUES (liste des valeurs**), (**liste d'autres valeurs**), (**liste d'encore d'autres valeurs**), …**

Exemple :

INSERT INTO *Ballon* VALUES (20, *'rouge'***), (**35, *'vert fluo'***), (**17, *'orange'***), (**28, *'mauve'***)**

Cet exemple est équivalent aux requêtes suivantes :

```
INSERT INTO Ballon VALUES(20, 'rouge')
INSERT INTO Ballon VALUES(35, 'vert fluo')
INSERT INTO Ballon VALUES(17, 'orange')
INSERT INTO Ballon VALUES(28, 'mauve')
```

13.2 Modification de données

UPDATE [LOW_PRIORITY] [IGNORE] [*nomBase*.] *nomTable*
SET *col_name1=expr1* [, *col_name2=expr2* ...]
SET *colonne1* = *expression1* | (*requête_SELECT*) | DEFAULT
[,*colonne2* = *expression2*...]
[WHERE (*condition*)]
[ORDER BY *listeColonnes*]
[LIMIT *nbreLimite*]

- `LOW_PRIORITY` indique que la modification est différée à la libération complète de la table (option à ne pas utiliser sur des tables `MyISAM`).
- `IGNORE` signifie que les éventuelles erreurs déclenchées suite aux modifications seront considérées en tant que *warning*s.
- La clause `SET` actualise une colonne en lui affectant une expression (valeur, valeur par défaut, calcul ou résultat d'une requête).
- La condition du `WHERE` filtre les lignes à mettre à jour dans la table. Si aucune condition n'est précisée, tous les enregistrements seront actualisés. Si la condition ne filtre aucune ligne, aucune mise à jour ne sera réalisée.
- `ORDER BY` indique l'ordre de modification des colonnes.

`LIMIT` spécifie le nombre maximum d'enregistrements à changer (par ordre de clé primaire croissante). Permet de n'appliquer la commande qu'aux **nbreLimite** premiers enregistrements satisfaisant la condition définie par **WHERE**.

Pour modifier un ou des enregistrement(s) d'une relation, il faut donc préciser un critère de sélection des enregistrements à modifier (clause **WHERE**), il faut aussi dire quels sont les attributs dont on va modifier la valeur et quelles sont ces nouvelles valeurs (clause **SET**).

Exemple :

```
UPDATE Personnes SET téléphone='0156281469' WHERE nom='Martin' AND prénom = 'Pierre'
```

Cet exemple modifie le numéro de téléphone de Martin Pierre.

Il est possible de modifier les valeurs d'autant d'attributs que la relation en contient.
Exemple pour modifier plusieurs attributs :

```
UPDATE Personnes SET téléphone='0156281469', fax='0156281812' WHERE id = 102
```

Pour appliquer la modification à tous les enregistrements de la relation, il suffit de ne pas mettre de clause **WHERE**.

Autre exemple :

```
UPDATE Enfants SET age=age+1
```

Il est donc possible de modifier la valeur d'un attribut relativement à sa valeur déjà existante.

13.3 Suppression de données

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM [nomBase.] nomTable
[WHERE (condition)]
[ORDER BY listeColonnes]
[LIMIT nbreLimite]
```

- `LOW_PRIORITY`, `IGNORE` et `LIMIT` ont la même signification que pour `UPDATE`.
- `QUICK` (pour les tables de type `MyISAM`) ne met pas à jour les index associés pour accélérer le traitement.
- La condition du `WHERE` sélectionne les lignes à supprimer dans la table. Si aucune condition n'est précisée, toutes les lignes seront détruites. Si la condition ne sélectionne aucune ligne, aucun enregistrement ne sera supprimé.
- `ORDER BY` réalise un tri des enregistrements qui seront effacés dans cet ordre.

Attention, la suppression est définitive !

Exemple :

```
DELETE FROM Personnes WHERE nom='Martin' AND prénom='Marc'
```

Pour vider une table de tous ces éléments, ne pas mettre de clause WHERE. Cela efface et recrée la table, au lieu de supprimer un à un chacun des enregistrements de la table (ce qui serait très long).
Exemple :

```
DELETE FROM Personnes
```

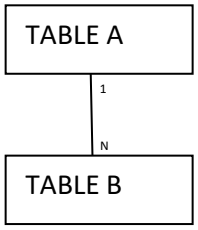
13.4 Exercices – insertion de données

Insérez en sql les personnes suivantes dans la table T_contact créée à l'exercice précédent :

titre	nom	prenom	Adresse	Numéro	CP	Ville	gsm	téléphone	email
Mr	AUBOISDORMANT	Abel	Rue des arbres	12	1300	Wavre	0479/794513	010/212121	auboisdormantabel@yahoo.fr
Dr	ZIEUVAIR	Bruno	Clos de l'Armoise	1323	1300	Wavre	0479/784513	010/212122	zieuvairbruno@yahoo.fr

14. Intégrité référentielle

La **normalisation** correspond au processus d'organiser ses données afin de limiter les redondances, divisant une table en plusieurs, et en les reliant entre elles par des clefs primaires et étrangères. L'objectif est d'isoler les données afin que l'ajout, l'effacement ou la modification d'un champ puisse se faire sur une seule table, et se propager au reste de la base par le biais des relations.



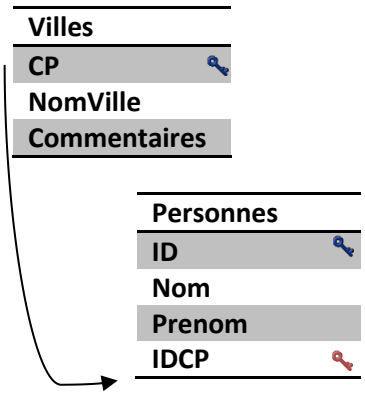
1. Clé primaire : identifiant unique et non nul
2. On ne peut pas ajouter un élément B pour un A inexistant
3. On ne peut pas supprimer un élément A s'il lui correspond un ou plusieurs éléments B

[CONSTRAINT *nomContrainte*] FOREIGN KEY [id] (*listeColonneEnfant*)
REFERENCES *nomTable* (*listeColonneParent*)
[ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
[ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]

- La cohérence du "fils" vers le "père" : on ne doit pas pouvoir insérer un enregistrement "fils" (ou modifier sa clé étrangère) rattaché à un enregistrement "père" inexistant. Il est cependant possible d'insérer un "fils" (ou de modifier sa clé étrangère) sans rattacher d'enregistrement "père", à la condition qu'il n'existe pas de contrainte `NOT NULL` au niveau de la clé étrangère.

- La cohérence du "père" vers le "fils" : on ne doit pas pouvoir supprimer un enregistrement "père" si un enregistrement "fils" y est encore rattaché. Il est possible de supprimer les "fils" associés (`DELETE CASCADE`), d'affecter la valeur nulle aux clés étrangères des "fils" associés (`DELETE SET NULL`) ou de répercuter une modification de la clé primaire du père (`UPDATE CASCADE` et `UPDATE SET NULL`).

Exemple :




```
CREATE TABLE Villes(
CP INT(4) PRIMARY KEY,
NomVille VARCHAR(80),
Commentaires TEXT
);
```

```
CREATE TABLE Personnes(
ID INT(11) auto_increment PRIMARY KEY,
Nom VARCHAR (40),
Prenom VARCHAR(40),
IDCP INT(4),
CONSTRAINT FK_CP FOREIGN KEY (IDCP)
REFERENCES Villes (CP));
```

```
INSERT INTO Villes VALUES(1300, 'Wavre','');
INSERT INTO Villes VALUES(1342, 'Limelette','');
INSERT INTO Villes VALUES(1340, 'Ottignies','');
INSERT INTO Villes VALUES(1348, 'LLN','');
INSERT INTO Villes VALUES(1330, 'Rixensart','');
```

```
INSERT INTO Personnes VALUES('','Martin','Bruno',1342); → ok
```

```
INSERT INTO Personnes VALUES('','Martin','Charles',1301); → erreur
```

#1452 - Cannot add or update a child row: a foreign key constraint fails

```
"CONSTRAINT FK_CP FOREIGN KEY (IDCP)
REFERENCES Villes (CP)
ON DELETE NO ACTION ON UPDATE NO ACTION"
```

Si tentative de suppression d'une ville → "foreign key constraint fails"

```
"CONSTRAINT FK_CP FOREIGN KEY (IDCP)
REFERENCES Villes (CP)
ON DELETE CASCADE ON UPDATE CASCADE"
```

Si tentative de suppression d'une ville → suppression de toutes les personnes
Si tentative de modification d'une ville (code postal par exemple) → mise à jour automatique dans la table personnes.

```
"CONSTRAINT FK_CP FOREIGN KEY (IDCP)
REFERENCES Villes (CP)
ON DELETE SET NULL"
```

Si tentative de suppression d'une ville → Affecter la valeur nulle dans la table personnes

14.1 Exercices – Analyse/ Intégrité référentielle

- Réalisez les tables qui permettraient de normaliser le tableau suivant. (L'idée est de ne pas "répéter" la fonction pour chaque employé – et par exemple de pouvoir ajouter des informations relatives à la fonction (barème, descriptif, etc.) sans devoir répéter ces informations pour chaque personne).

N°	Nom	Prénom	Fonction
1	DURANT	Louis	Commercial
2	DURAND	Pierrette	Webmaster
3	MARÉCHAL	Juan	Webmaster
4	MICHEL	Louisette	Commercial
5	TLUAUP	Yves	PDG
6	BACH	Gérard	Graphiste
7	DELLILE	Georgette	Graphiste
8	DELAGRANGE	Maurice	Comptable

- Réalisez les tables qui contiendraient toutes les informations relatives aux avions ainsi qu'à leurs vols

AVIONS

AviID	AviModel	AviNombreDePlaces	AviLocalite
1	A300	300	NICE
2	A310	300	NICE
3	B707	250	PARIS
4	A300	280	LYON
5	CONCORDE	160	NICE
6	B747	460	PARIS
7	B707	250	PARIS
8	A310	300	TOULOUSE
9	MERCURE	180	LYON
10	CONCORDE	160	PARIS

VOLS

VolID	VolDate	VolDestionationDepart	VolDestinationArrivé	VolHeureDepart	VoleHeureArrivé	VolAviID
1	2013-10-30	NICE	TOULOUSE	11h00	12h30	1
2	2013-10-30	PARIS	TOULOUSE	17h00	18h30	8
3	2013-10-30	TOULOUSE	LYON	14h00	16h00	1
4	2013-10-30	TOULOUSE	LYON	18h00	20h00	3
5	2013-10-30	PARIS	NICE	06h45	08h15	1
6	2013-10-30	LYON	NICE	11h00	12h00	2
7	2013-10-30	PARIS	LYON	08h00	09h00	4
8	2013-10-30	NICE	PARIS	07h15	08h45	4
9	2013-10-31	NANTES	LYON	09h00	15h30	8
10	2013-10-31	NICE	PARIS	12h15	13h45	2
11	2013-10-31	PARIS	LYON	15h00	16h00	2
12	2013-10-31	LYON	NANTES	16h30	20h00	2
13	2013-10-31	NICE	LENS	11h00	14h00	5
14	2013-10-31	LENS	PARIS	15h00	16h00	5
15	2013-10-31	PARIS	TOULOUSE	17h00	18h00	9
16	2013-10-31	PARIS	TOULOUSE	18h00	19h00	5

- Dans le service achats, on a besoin de connaître pour chaque article quels sont les fournisseurs qui fournissent ledit article. Un article peut être fourni par plusieurs fournisseurs différents et un fournisseur fournit plusieurs articles. Réalisez le diagramme de la base de données avec l'indication des clefs primaires.
- « Gestion d'une association d'anciens »

L'association des anciens étudiants d'une école souhaitent informatiser la gestion de son fichier des « anciens ».
Pour tout ancien étudiant, elle détient son nom, prénom et adresse ainsi que son numéro d'inscription à l'école et elle connait les diplômes qu'il a obtenus, ainsi que leurs dates d'obtention.

Elle essaie également de connaître la liste des entreprises qui les ont employé par le passé et celles dans laquelle ils sont actuellement employé ainsi que et la fonction ou les fonctions qu'ils y occupent ou qu'ils ont occupés. Ils peuvent bien sur avoir exercés plusieurs fonctions différentes dans plusieurs sociétés différentes.

En vue de produire des statistiques relatives à l'emploi des anciens étudiants, il serait également intéressant de connaitre la durée de chaque emploi presté.

Réalisez le schéma de la base de données ainsi que les relations entre les tables. (N'oubliez pas de bien précisez les clés primaires et clés étrangères.) Reproduisez le code SQL de la création de la table « ***Anciens*** » (uniquement celle-là) avec les clés primaires (*autoincrement*) et les clés étrangères s'il y en a.

- Fleuves

Réalisez le schéma relationnel de la BD suivante, en définissant clairement les clés primaires ainsi que les clés étrangères :

« On souhaite créer une base de données destinée à la gestion des pays et de ces fleuves. Un pays est connu par un nom, son identifiant, une superficie, un nombre d'habitants et la liste des fleuves qui le traversent. Un fleuve est connu par son identifiant, son nom, sa longueur, le nom du pays dans lequel il prend sa source, la liste des pays qu'il traverse, la distance parcourue dans chacun des pays. »

15. Mise à jour d'une base de données

15.1 Modification d'une table

ALTER TABLE

Il est possible de modifier la définition d'une table par la suite, à tout moment par la commande **ALTER TABLE**.
Voici ce qu'il est possible de réaliser :
- ajouter/supprimer un attribut
- créer/supprimer une clé primaire
- ajouter une contrainte d'unicité (interdire les doublons)
- changer la valeur par défaut d'un attribut
- changer totalement la définition d'un attribut
- changer le nom de la relation
- ajouter/supprimer un index

15.1.1 Ajouter un attribut

Syntaxe :

```
ALTER TABLE relation ADD definition [ FIRST | AFTER attribut]
```

Ajoutons l'attribut *fax* qui est une chaîne représentant un nombre de 10 chiffres:

```
ALTER TABLE Personnes ADD fax DECIMAL(10,0)
```

Nous aurions pu forcer la place où doit apparaître cet attribut. Pour le mettre en tête de la liste des attributs de la relation, il faut ajouter l'option **FIRST** en fin de commande. Pour le mettre après l'attribut *'téléphone'*, il aurait fallu ajouter **AFTER** *'téléphone'*.
Note : il **ne** peut **pas** déjà avoir dans la relation un attribut du même nom !

15.1.2 Supprimer un attribut

Attention, supprimer un attribut implique la suppression des valeurs qui se trouvent dans la colonne qui correspond à cet attribut.
Syntaxe :

```
ALTER TABLE relation DROP attribut
```

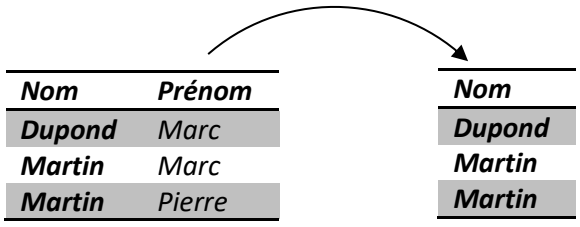
Exemple :

```
ALTER TABLE Personnes DROP 'prenom'
```

La suppression d'un attribut peut incidemment provoquer des erreurs sur les contraintes clé primaire (**PRIMARY KEY**) et unique (**UNIQUE**).

```
CREATE TABLE Personne (
        id SMALLINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
        nom VARCHAR(40),
        'prénom' VARCHAR(40),
        adresse TINYTEXT,
        'téléphone' DECIMAL(10,0),
        UNIQUE(nom,'prénom')
)
```

ALTER TABLE *Personnes* DROP *'prénom'*



Nom	***Prénom***
Dupond	*Marc*
Martin	*Marc*
Martin	*Pierre*

Nom
Dupond
Martin
Martin

Refus d'opérer la suppression, car cela contredirait la contrainte d'unicité qui resterait sur l'attribut *nom*.

15.1.3 Créer une clé primaire

La création d'une clé primaire n'est possible qu'en l'absence de clé primaire dans la relation.
Syntaxe :

```
ALTER TABLE relation ADD PRIMARY KEY (attribut)
```

Exemple :

```
ALTER TABLE Personnes ADD PRIMARY KEY (nom,prenom)
```

15.1.4 Supprimer une clé primaire

Comme une clé primaire est unique, il n'y a aucune ambiguïté lors de la suppression.
Syntaxe :

```
ALTER TABLE relation DROP PRIMARY KEY
```

Exemple :

```
ALTER TABLE Personnes DROP PRIMARY KEY
```

S'il n'y a aucune clé primaire lorsque cette commande est exécutée, aucun message d'erreur ne sera généré, la commande sera simplement ignorée.

15.1.5 Ajout d'une contrainte d'unicité

Il est possible (facultatif) de donner un nom à la contrainte.
Cette contrainte peut s'appliquer à plusieurs attributs.
Si les valeurs déjà présentes dans la relation sont en contradiction avec cette nouvelle contrainte, alors cette dernière ne sera pas appliquée et une erreur sera générée.
Syntaxe :

```
ALTER TABLE relation ADD UNIQUE [contrainte] (attributs)
```

Exemple pour interdire tout doublon sur l'attribut *fax* de la relation ***Personnes*** :

```
ALTER TABLE Personnes ADD UNIQUE u_fax (fax)
```

Autre exemple fictif :

```
ALTER TABLE Moto ADD UNIQUE u_coul_vitre (couleur,vitre)
```

15.1.6 Changer la valeur par défaut d'un attribut

Pour changer ou supprimer la valeur par défaut d'un attribut.
Attention aux types qui n'acceptent pas de valeur par défaut (**BLOB** et **TEXT**).
Syntaxe :

```
ALTER TABLE relation ALTER attribut { SET DEFAULT valeur | DROP DEFAULT }
```

Changer sa valeur par défaut :

```
ALTER TABLE Personnes ALTER 'téléphone' SET DEFAULT '9999999999'
```

Supprimer sa valeur par défaut :

```
ALTER TABLE Personnes ALTER 'téléphone' DROP DEFAULT
```

Le changement ou la suppression n'affecte en rien les enregistrements qui ont eu recours à cette valeur lors de leur insertion.

15.1.7 Changer la définition d'un attribut

Pour changer la définition de l'attribut sans le renommer :

```
ALTER TABLE relation MODIFY attribut definition_relative
```

Exemple 1 :

```
ALTER TABLE Personnes MODIFY fax VARCHAR(14)
```

Pour changer sa définition en le renommant :

```
ALTER TABLE relation CHANGE attribut definition_absolue
```

Exemple 2 :

```
ALTER TABLE Personnes CHANGE fax num_fax VARCHAR(14)
```

Attention, si le nouveau type appliqué à l'attribut est incompatible avec les valeurs des enregistrements déjà présents dans la relation, alors les valeurs risquent d'être modifiées ou remises à zéro !

15.1.8 Changer le nom d'une relation

Syntaxe :

```
ALTER TABLE relation RENAME nouveau_nom
```

Exemple :

```
ALTER TABLE Personnes RENAME Carnet
```

Cela consiste à renommer la table, et donc le fichier qui la stocke.

15.1.9 Ajouter un index

Une table ne peut comporter que 32 index.
Et un index ne peut porter que sur 16 attributs maximum à la fois.
Syntaxe :

```
ALTER TABLE relation ADD INDEX index (attributs)
```

Exemple :

```
ALTER TABLE Personnes ADD INDEX nom_complet (nom,prénom)
```

Dans cet exemple, on a ajouté à la relation ***Personnes*** un index que l'on nomme *nom_complet* et qui s'applique aux deux attributs '*nom*' et '*prénom*'. Ainsi, les recherches et les tris sur les attributs '*nom*' et '*prénom*' seront grandement améliorés. Car un index apporte les changements sous-jacents permettant d'optimiser les performances du serveur de base de données.

Remarques :

Après la suppression de grandes parties d'une table contenant des index, les index des tuples supprimés sont conservés, rallongeant d'autant les sélections. Pour supprimer ces index obsolètes et vider les "trous", il faut l'optimiser.

Syntaxe :

```
OPTIMIZE TABLE Relation
```

Exemple :

```
OPTIMIZE TABLE Personnes
```

15.1.10 Supprimer un index

Syntaxe :

```
ALTER TABLE relation DROP INDEX index
```

Exemple :

```
ALTER TABLE Personnes DROP INDEX nom_complet
```

Cet exemple permet de supprimer l'index nommé *nom_complet* de la relation ***Personnes***.

15.2 Suppression d'une table

```
DROP TABLE
```

La commande **DROP TABLE** prend en paramètre le nom de la table à supprimer. Toutes les données qu'elle contient sont supprimées et sa définition aussi.

Syntaxe :

```
DROP TABLE relation
```

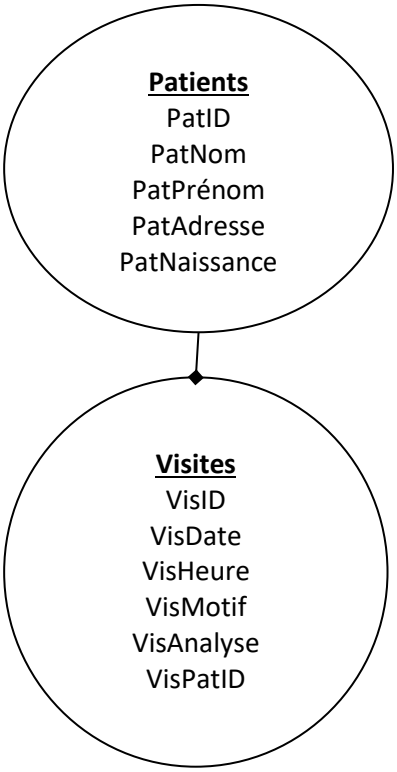
Exemple :

```
DROP TABLE Personnes
```

15.3 Exercices – mise à jour d'une base de données

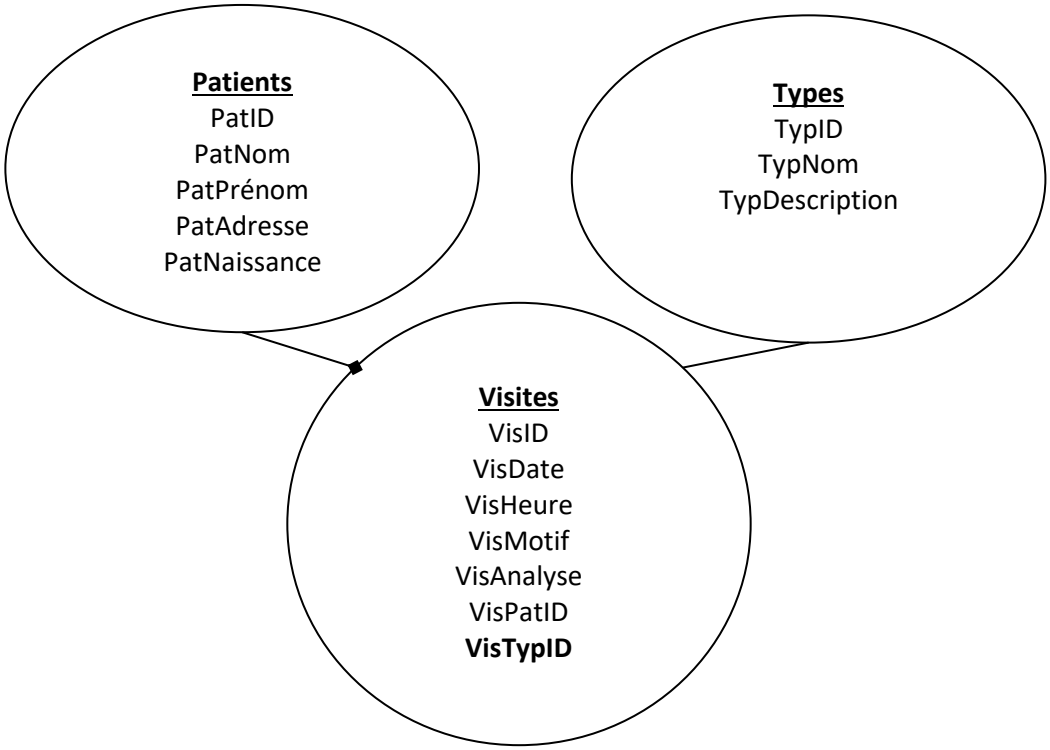
Gestion des visites de patients chez le médecin.

Créez la base de données "clinique" avec les tables suivantes



Ajoutez une contrainte d'unicité sur le champ "date" et "heure" (un patient ne peut pas avoir un rendez-vous avec un autre patient !)

Ajoutez la table "Types", et modifiez la table "Visites" pour lui ajouter la clé étrangère "VisTypID" :



Ajoutez un index sur le nom et le prénom de la table patients

16 Interrogation de base de données

16.1 Requêtes de sélection

Syntaxe générale :

```
SELECT [ DISTINCT ] attributs
        [ INTO OUTFILE fichier ]
        [ FROM relation ]
        [ WHERE condition ]
        [ GROUP BY attributs  ]
        [ HAVING condition ]
        [ ORDER BY attributs [ ASC | DESC ]]
        [ LIMIT [a,] b ]
```

Nom	Description
SELECT	Spécifie les attributs dont on souhaite connaître les valeurs.
DISTINCT	Permet d'ignorer les doublons de ligne de résultat.
INTO OUTFILE	Spécifie le fichier sur lequel effectuer la sélection.
FROM	Spécifie le ou les relations sur lesquelles effectuer la sélection.
WHERE	Définit le ou les critères de sélection sur des attributs.
GROUP BY	Permet de grouper les lignes de résultats selon un ou des attributs.
HAVING	Définit un ou des critères de sélection sur des ensembles de valeurs d'attributs après groupement.
ORDER BY	Permet de définir l'ordre (ASCendant par défaut ou DESCendant) dans l'envoi des résultats.
LIMIT	Permet de limiter le nombre de lignes du résultat

16.1.1 Simple

Projection (vue) : on ne sélectionne qu'un ou plusieurs attributs d'une relation (on ignore les autres).

Personnes

Nom	Prénom	Adresse	Téléphone
Martin	Pierre	7 Place des Ronds	0258941236
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soulards	0123456789

On projette la table Personnes sur les colonnes nom et prénom

SELECT nom, prénom
FROM Personnes

Nom	Prénom
Martin	Pierre
Dupond	Jean
Dupond	Marc

Pour sélectionner tous les enregistrements d'une relation :

```
SELECT * FROM relation
```

Pour sélectionner toutes les valeurs d'un seul attribut :

```
SELECT attribut FROM relation
```

Pour éliminer les doublons :

```
SELECT DISTINCT attribut FROM relation
```

16.1.2 Avec filtre

Le filtre s'effectue sur un ou plusieurs attributs. La relation résultante a la même structure tandis que le contenu résultera des critères de sélection qui portent sur les valeurs des attributs.

Personnes

Nom	Prénom	Adresse	Téléphone
Martin	Pierre	7 Place des Ronds	0258941236
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soulards	0123456789

SELECT *
FROM Personnes
WHERE nom = ''Dupond''

On ne sélectionne que les tuples dont l'attribut nom est égal à « Dupond »

Nom	Prénom	Adresse	Téléphone
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soulards	0123456789

Par exemple, extraction de votre base de données de la liste des personnes de votre carnet d'adresse qui vivent à Paris.

```
SELECT nom,prénom FROM Personnes WHERE adresse LIKE '%paris%'
```

16.1.3 Avec tri

Le tri s'effectue sur un ou plusieurs attributs, dans l'ordre croissant ou décroissant. La relation résultante a la même structure et le même contenu que la relation de départ.
La commande du tri est la clause **`ORDER BY`** et à défaut de spécifications **`DESC`** (Descending) ou **`ASC`** (Ascending), le tri est croissant.

Personnes

Nom	Prénom	Adresse	Téléphone
Martin	Pierre	7 Place des Ronds	0258941236
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soulards	0123456789

SELECT *
FROM Personnes
ORDER BY prénom ASC

On sélectionne tous les tuples en les triant par ordre alphabéthique de « prénom »

Nom	Prénom	Adresse	Téléphone
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soulards	0123456789
Martin	Pierre	7 Place des Ronds	0258941236

Pour trier les valeurs en ordre croissant :

```
SELECT DISTINCT attribut FROM relation ORDER BY attribut ASC
```

Pour se limiter aux **num** premiers résultats :

```
SELECT DISTINCT attribut FROM relation ORDER BY attribut ASC LIMIT num
```

Pour ne sélectionner que ceux qui satisfont à une condition :

```
SELECT DISTINCT attribut FROM relation WHERE condition ORDER BY attribut ASC LIMIT num
```

16.1.4 Exercices – Sélection simple/filtre/tri

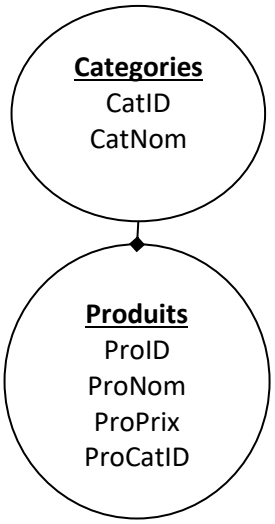
Soit la table "gens"

Nom	Prenom	Age
Dupond	Pierre	24
Martin	Marc	48
Dupont	Jean	51
Martin	Paul	36
Dupond	Lionel	68
Chirac	Jacques	70

1) Sélectionnez toute la table:
2) Sélectionnez uniquement les "noms" des personnes de la table "gens":
3) Sélectionnez uniquement les "noms" des personnes de la table "gens" en supprimant les doublons:
4) Sélectionnez uniquement les "noms" par ordre alphabétique, des personnes de la table "gens" en supprimant les doublons:
5) Affichez uniquement les deux premiers résultats de la dernière requête:
6) Affichez uniquement les deux premiers résultats de la dernière requête sauf si le nom est égal à "Chirac":

16.1.5 Multiple

Soit un système de gestion catégories/produits :




```
CREATE TABLE Categories (
   CatID int(3) PRIMARY KEY auto_increment,
   CatNom varchar(30)
);

CREATE TABLE Produits (
   ProID int(3) PRIMARY KEY auto_increment,
   ProNom varchar(255),
   ProPrix int(7),
   ProCatID int(3),
  CONSTRAINT FK_Cat FOREIGN KEY(ProCatID) REFERENCES Categories(CatID)

);


INSERT INTO Categories(CatNom) VALUES( 'Immobilier');
INSERT INTO Categories(CatNom)  VALUES( 'Services');
```

```
INSERT INTO Categories(CatNom)  VALUES( 'Auto');
INSERT INTO Categories(CatNom)  VALUES( 'Cinéma');
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Maison', '150000', '1') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Appartement', '60000', '1') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Yourthe', '10000', '1') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Nettoyage', '' , '2') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Repassage', '' , '2') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Brad Pitt', '1000' , '4') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Mel Gibson', '500' , '4') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Jean Paul Belmondo', '10' , '4') ;
```

SELECT * FROM Produits

SELECT * FROM Categories

CatID	CatNom
1	Immobilier
2	Services
3	Auto
4	Cinéma

ProID	ProNom	ProPrix	ProCatID
1	Maison	150000	1
2	Appartement	60000	1
3	Yourthe	10000	1
4	Nettoyage	0	2
5	Repassage	0	2
8	Brad Pitt	1000	4
9	Mel Gibson	500	4
10	Jean Paul Belmondo	10	4
12	Un machin Bidule	100	*NULL*

16.1.6.1 Requêtes imbriquées :

Lorsqu'il y a plusieurs tables, et que nous voulons récupérer par exemple les "produits" de la catégorie "Immobilier", le programmeur non initié correctement va faire minimum deux requêtes, une première pour récupérer l'id du produits dans la table Produits :

```
SELECT CatID FROM Categories  WHERE CatNom='Immobilier';
```

➔ Retourne la valeur "1"

Puis une deuxième pour récupérer les produits :

```
SELECT * FROM Produits WHERE ProCatID='1';
```

Le résultat de la 1ère requête peut être directement injecté dans la deuxième :

```
SELECT * FROM Produits WHERE ProCatID=( SELECT CatID FROM Categories  WHERE CatNom='Immobilier') ;
```

16.1.6.2 Requêtes sélection multi-tables:

L'inconvénient de la méthode précédente est que le SGBD va quand même exécuter deux requêtes. La solution est d'utiliser une requête de sélection multi-tables :

```
SELECT *
FROM Categories, Produits
WHERE Categories.CatID = Produits.ProCatID
AND Categories.CatNom="Immobilier"
```

Remarque : L'instruction AS permet de définir un alias, ce qui évite de réécrire le nom de la table au complet.

```
SELECT *
FROM Categories AS c, Produits AS p
WHERE c.CatID = p.ProCatID
AND c.CatNom="Immobilier"
```

Au lieu des requêtes multi-tables, la clause INNER JOIN a fait son apparition avec la version 2 de SQL, parce que le besoin s'était fait sentir de préciser à quel type de jointure appartenait une relation. Les puristes préconiseront donc le "Inner join" (voir point 7.1.9).

16.1.6 Avec regroupement

L'instruction "Group by" permet de regrouper les lignes selon un critère.

```
SELECT ...
FROM nom_table
[WHERE condition]
GROUP BY nom_colonne;
```

Soit la table "animal" suivante :

```
CREATE TABLE Animal (
id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
espece VARCHAR(40) NOT NULL,
sexe CHAR(1),
date_naissance DATETIME NOT NULL,
nom VARCHAR(30),
commentaires TEXT,
PRIMARY KEY (id)
);
```

id	espece	sexe	date_naissance	nom	commentaires
1	chien	F	2008-02-20 15:45:00	Canaille	*NULL*
2	chien	F	2009-05-26 08:54:00	Cali	*NULL*
3	chien	F	2007-04-24 12:54:00	Rouquine	*Mordille parfois*
4	chien	F	2009-05-26 08:56:00	Fila	*NULL*
5	chien	F	2008-02-20 15:47:00	Anya	*NULL*
6	chien	F	2009-05-26 08:50:00	Louya	*NULL*
7	chien	F	2008-03-10 13:45:00	Welva	*Dangereux*
8	chien	F	2007-04-24 12:59:00	Zira	*NULL*
9	chien	F	2009-05-26 09:02:00	Java	*NULL*
10	chien	M	2007-04-24 12:45:00	Balou	*NULL*
11	chien	M	2008-03-10 13:43:00	Pataud	*NULL*
12	chien	M	2007-04-24 12:42:00	Bouli	*NULL*
13	chien	M	2009-03-05 13:54:00	Zoulou	*NULL*
14	chien	M	2007-04-12 05:23:00	Cartouche	*NULL*
15	chien	M	2006-05-14 15:50:00	Zambo	*NULL*
16	chien	M	2006-05-14 15:48:00	Samba	*NULL*
17	chien	M	2008-03-10 13:40:00	Moka	*NULL*
18	chien	M	2006-05-14 15:40:00	Pilou	*NULL*
19	chat	M	2009-05-14 06:30:00	Fiero	*NULL*
20	chat	M	2007-03-12 12:05:00	Zonko	*NULL*
21	chat	M	2008-02-20 15:45:00	Filou	*NULL*
22	chat	M	2007-03-12 12:07:00	Farceur	*NULL*
23	chat	M	2006-05-19 16:17:00	Caribou	*NULL*
24	chat	M	2008-04-20 03:22:00	Capou	*NULL*
25	chat	M	2006-05-19 16:56:00	Raccou	Pas de queue depuis la naissance

L'instruction suivante

```
SELECT espece, COUNT(*) AS nb_animaux
FROM Animal
GROUP BY espece;
```

Retournera :

espece	nb_animaux
chat	7
chien	18

Si on ne souhaite prendre que les mâles :

```
SELECT espece, COUNT(*) AS nb_males
FROM Animal
WHERE sexe = 'M'
GROUP BY espece;
```

Retournera :

espece	nb_males
chat	7
chien	9

Remarque :

Lorsque l'on fait un groupement dans une requête, avec GROUP BY, on ne peut sélectionner que deux types d'éléments dans la clause SELECT :

- une ou des colonnes **ayant servi de critère** pour le regroupement ;
- **une fonction d'agrégation** (agissant sur n'importe quelle colonne).

16.1.6.1 MySQL gère-t-il le GROUP BY comme les autres SGBD ?

Non, MySQL applique une optimisation spécifique qui constitue une "extension de la norme" et est donc en contradiction avec celle-ci.
MySQL est un SGBD extrêmement permissif. Dans certains cas, c'est bien pratique, mais c'est toujours dangereux.
Et notamment en ce qui concerne GROUP BY, MySQL ne sera pas perturbé si vous sélectionnez une colonne qui n'est pas dans les critères de regroupement. Reprenons la requête qui sélectionne la colonne date_naissance alors que le regroupement se fait sur la base de l'espèce. Cette requête ne respecte pas la norme SQL, et n'a aucun sens. La plupart des SGBD vous renverront une erreur si vous tentez de l'exécuter.

```
SELECT espece, COUNT(*) AS nb_animaux, date_naissance
FROM Animal
GROUP BY espece;
```

Retournera

espece	nb_animaux	date_naissance
chat	7	2009-05-14 06:30:00
chien	18	2008-02-20 15:45:00

MySQL a tout simplement pris n'importe quelle valeur parmi celles du groupe pour la date de naissance.
Soyez donc très prudents lorsque vous utilisez GROUP BY. Vous faites peut-être des requêtes qui n'ont aucun sens, et MySQL ne vous en avertira pas !

16.1.7.2 Regroupement sur plusieurs critères :

```
SELECT espece, sexe, COUNT(*) as nb_animaux
FROM Animal
GROUP BY espece, sexe;
```

espece	sexe	nb_animaux
chat	M	7
chien	F	9
chien	M	9

16.1.7 Champs calculés

Un champ calculé dans une requête est le résultat d'un calcul ou d'une fonction sur un des champs de la table.
Exemple :

```
SELECT MIN(ProPrix) AS Minimun, MAX(ProPrix) AS Maximum FROM Produits
```

Minimun	Maximum
0	150000

```
SELECT avg( ProPrix ) FROM `produits`
```

avg(ProPrix)
27688.7500

Calcul de 21% sur le prix :

```
SELECT ProNom, ProPrix*1.21 FROM produits
```

ProNom	ProPrix*1.21
Maison	181500.00
Appartement	72600.00
Yourthe	12100.00
Nettoyage	0.00
Repassage	0.00
Brad Pitt	1210.00
Mel Gibson	605.00
Jean Paul Belmondo	12.10
Un machin Bidule	121.00

16.1.8 Fonctions de MySQL

Ces fonctions sont à ajouter à vos requêtes dans un SELECT, WHERE, GROUP BY ou encore HAVING.
Vous avez à votre disposition :

- les parenthèses **( )**,
- les opérateurs arithmétiques (**+**, **-**, *****, **/**, **%**),
- les opérateurs binaires (**<**, **<<** , **>**, **>>**, **|**, **&**),
- les opérateurs logiques qui retournent **0** (faux) ou **1** (vrai) (**AND**, **OR**, **NOT**, **BETWEEN**, **IN**),
- les opérateurs relationnels (**<**, **<=**, **=**, **>**, **>=**, **<>**).

Les opérateurs et les fonctions peuvent êtres composés entre eux pour donner des expressions très complexes.

16.1.8.1 Quelques exemples

```
SELECT nom
FROM produits
WHERE prix <= 100.5
```

Liste du nom des produits dont le prix est inférieur ou égal à 100.5 EUR.

```
SELECT nom,prénom
FROM élèves
WHERE age BETWEEN 12 AND 16
```

Liste des nom et prénom des élèves dont l'âge est compris entre 12 et 16 ans.

```
SELECT modèle
FROM voitures
WHERE couleur IN ('rouge', 'blanc', 'noir')
```

Liste des modèles de voiture dont la couleur est dans la liste : rouge, blanc, noir.

```
SELECT modèle
FROM voitures
WHERE couleur NOT IN ('rose', 'violet')
```

Liste des modèles de voiture dont la couleur n'est pas dans la liste : rose, violet.

16.1.8.2 Fonctions de comparaison de chaînes

- Le mot clé **LIKE** permet de comparer deux chaînes.
- Le caractère '**%**' est spécial et signifie : 0 ou plusieurs caractères.
- Le caractère '**_**' est spécial et signifie : 1 seul caractère, n'importe lequel.

L'exemple suivant permet de rechercher tous les clients dont le prénom commence par 'Jean', cela peut être 'Jean-Pierre', etc… :

```
SELECT nom
FROM clients
WHERE prénom LIKE 'Jean%'
```

Pour utiliser les caractères spéciaux ci-dessus en leur enlevant leur fonction spéciale, il faut les faire précéder de l'antislash : '****'.
Exemple, pour lister les produits dont le code commence par la chaîne '_XE' :

```
SELECT *
FROM produit
WHERE code LIKE '\_XE%'
```

16.1.8.3 Fonctions mathématiques

Fonction	Description
ABS(x)	Valeur absolue de X.
SIGN(x)	Signe de X, retourne -1, 0 ou 1.
FLOOR(x)	Arrondi à l'entier inférieur.
CEILING(x)	Arrondi à l'entier supérieur.
ROUND(x)	Arrondi à l'entier le plus proche.
EXP(x), LOG(x), SIN(x), COS(x), TAN(x), PI()	fonctions mathématiques de base…
POW(x,y)	Retourne X à la puissance Y.
RAND(), RAND(x)	Retourne un nombre aléatoire entre 0 et 1.0 Si x est spécifié, entre 0 et X
TRUNCATE(x,y)	Tronque le nombre X à la Yème décimale.

```
SELECT nom
FROM filiales
WHERE SIGN(ca) = -1
ORDER BY RAND()
```

Cet exemple affiche dans un ordre aléatoire le nom des filiales dont le chiffre d'affaire est négatif.
A noter que : SIGN(ca) = -1 ⇔ ca < 0

16.1.8.4 Fonctions de chaînes

Fonction	Description
TRIM(x)	Supprime les espaces de début et de fin de chaîne.
LOWER(x)	Converti en minuscules.
UPPER(x)	Converti en majuscules.
LONGUEUR(x)	Retourne la taille de la chaîne.
LOCATE(x,y)	Retourne la position de la dernière occurrence de x dans y. Retourne 0 si x n'est pas trouvé dans y.
CONCAT(x,y,…)	Concatène ses arguments.
SUBSTRING(s,i,n)	Retourne les n derniers caractères de s en commençant à partir de la position i.
SOUNDEX(x)	Retourne une représentation phonétique de x.

```
SELECT UPPER(nom)
FROM clients
WHERE SOUNDEX(nom) = SOUNDEX('Dupond')
```

On affiche en majuscules le nom de tous les clients dont le nom ressemble à 'Dupond'.

16.1.8.5 Fonctions de dates et heures

Fonction	Description
NOW()	Retourne la date et heure du jour.
TO_DAYS(x)	Conversion de la date X en nombre de jours depuis le 1er janvier 1970.
DAYOFWEEK(x)	Retourne le jour de la semaine de la date x sous la forme d'un index qui commence à 1 (1=dimanche, 2=lundi…)
DAYOFMONTH(x)	Retourne le jour du mois (entre 1 et 31).
DAYOFYEAR(x)	Retourne le jour de l'année (entre 1 et 366).
SECOND(x), MINUTE(x), HOUR(x), MONTH(x), YEAR(x), WEEK(x)	Retournent respectivement les secondes, minutes, heures, mois, année et semaine de la date.
DATEDIFF(expr,expr2)	retourne le nombre de jours entre la date de début expr et la date de fin expr2

Cet exemple affiche le titre des articles parus il y a moins de 30 jours.

```
SELECT titre
FROM article
WHERE (TO_DAYS(NOW()) – TO_DAYS(parution)) < 30
```

Celui-ci affiche le nom et prénom des personnes dont l'année de la date est "2002"

```
SELECT Nom, Prénom, Date
FROM Personnes
WHERE Year([Date])=2002;
```

16.1.9.6 Fonctions à utiliser dans les GROUP BY

Fonction	Description
COUNT([DISTINCT]x,y,…)	Décompte des tuples du résultat par projection sur le ou les attributs spécifiés (ou tous avec '*'). L'option DISTINCT élimine les doublons.
MIN(x), MAX(x), AVG(x), SUM(x)	Calculent respectivement le minimum, le maximum, la moyenne et la somme des valeurs de l'attribut X.

```
SELECT DISTINCT model
FROM voiture
GROUP BY model
HAVING COUNT(couleur) > 10
```

Ici on affiche les modèles de voitures qui proposent un choix de plus de 10 couleurs.

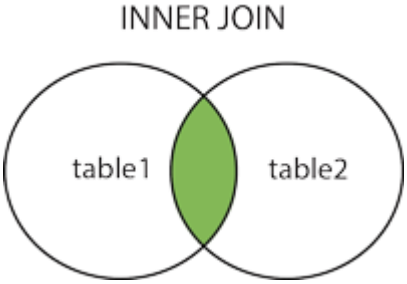
```
SELECT COUNT(*)
FROM client
```

Affichage du nombre de clients.

```
SELECT DISTINCT produit.nom, SUM(vente.qt * produit.prix) AS total
FROM produit, vente
WHERE produit.id = vente.produit_idx
GROUP BY produit.nom
ORDER BY total
```

Classement des produits par la valeur totale vendue.

16.1.9 Les jointures internes



Les jointures **internes**[4] (*inner joins*) ; à chaque ligne de la 1er table est associée toute ligne de la 2ème table vérifiant la condition.

Par exemple, pour obtenir les tuples des catégories qui contiennent des produits.

```
SELECT *
FROM categories
INNER JOIN produits ON categories.CatID = produits.ProCatID
```

CatID	CatNom	ProID	ProNom	ProPrix	ProCatID
1	Immobilier	1	Maison	150000	1
1	Immobilier	2	Appartement	60000	1
1	Immobilier	3	Yourthe	10000	1
2	Services	4	Nettoyage	0	2
2	Services	5	Repassage	0	2
4	Cinéma	8	Brad Pitt	1000	4
4	Cinéma	9	Mel Gibson	500	4
4	Cinéma	10	Jean Paul Belmondo	10	4

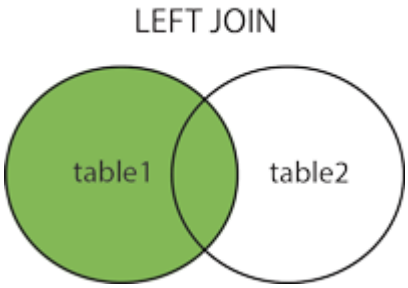
Ce premier type de jointure complexe apporte finalement peu de nouveautés puisque le "INNER JOIN" fonctionne exactement de la même manière qu'une jointure simple (avec la clause WHERE). Son principal intérêt est d'apporter une certaine lisibilité et de mieux distinguer les jointures internes (INNER) des jointures externes (OUTER).

16.1.10 Les jointures externes

La jointure externe (*outer join*), la plus compliquée, qui favorise une table (dite "dominante") par rapport à l'autre (dite "subordonnée"). Les lignes de la table dominante sont retournées même si elles ne satisfont pas aux conditions de jointure.
Pour bien comprendre les jointures externes :

[4] Quand le type de jointure est omis, c'est automatiquement une jointure interne (inner)

7.1.11.1 left outer



Une jointure interne est d'abord effectuée, puis y est ajoutée toute ligne de la 1ère table sans concordance avec la 2ème table et alors complétée par des "null".

Par exemple pour obtenir toutes les catégories, chacune avec tous les produits s'il y en existe.

```
SELECT *
FROM categories
LEFT OUTER JOIN produits ON categories.CatID = produits.ProCatID
```

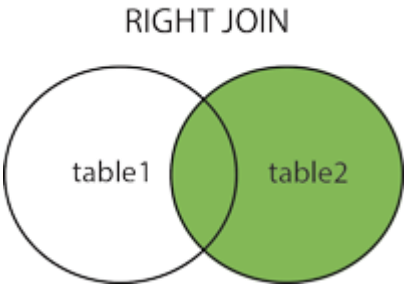
CatID	CatNom	ProID	ProNom	ProPrix	ProCatID
1	Immobilier	1	Maison	150000	1
1	Immobilier	2	Appartement	60000	1
1	Immobilier	3	Yourthe	10000	1
2	Services	4	Nettoyage	0	2
2	Services	5	Repassage	0	2
3	Auto	*NULL*	*NULL*	*NULL*	*NULL*
4	Cinéma	8	Brad Pitt	1000	4
4	Cinéma	9	Mel Gibson	500	4
4	Cinéma	10	Jean Paul Belmondo	10	4

Par exemple pour obtenir tous les produits chacun avec sa catégorie s'il y en a une.

```
SELECT *
FROM produits
LEFT OUTER JOIN categories ON produits.ProCatID = categories.CatID
```

ProID	ProNom	ProPrix	ProCatID	CatID	CatNom
1	Maison	150000	1	1	Immobilier
2	Appartement	60000	1	1	Immobilier
3	Yourthe	10000	1	1	Immobilier
4	Nettoyage	0	2	2	Services
5	Repassage	0	2	2	Services
8	Brad Pitt	1000	4	4	Cinéma
9	Mel Gibson	500	4	4	Cinéma
10	Jean Paul Belmondo	10	4	4	Cinéma

16.1.10.2 right outer



Une jointure interne est d'abord effectuée, puis y est ajoutée toute ligne de la 2ème table sans concordance avec la 1ère table et alors complétée par des "null".

Par exemple pour obtenir toutes les catégories, chacune avec tous les produits s'il y en existe.

```
SELECT *
FROM produits
RIGHT OUTER JOIN categories ON produits.ProCatID = categories.CatID
```

ProID	ProNom	ProPrix	ProCatID	CatID	CatNom
1	Maison	150000	1	1	Immobilier
2	Appartement	60000	1	1	Immobilier
3	Yourthe	10000	1	1	Immobilier
4	Nettoyage	0	2	2	Services
5	Repassage	0	2	2	Services
NULL	*NULL*	*NULL*	*NULL*	3	Auto
8	Brad Pitt	1000	4	4	Cinéma
9	Mel Gibson	500	4	4	Cinéma
10	Jean Paul Belmondo	10	4	4	Cinéma

Par exemple pour obtenir tous les produits chacun avec sa catégorie s'il y en a une.

```
SELECT *
FROM categories
RIGHT OUTER JOIN produits ON categories.CatID = produits.ProCatID
```

CatID	CatNom	ProID	ProNom	ProPrix	ProCatID
1	Immobilier	1	Maison	150000	1
1	Immobilier	2	Appartement	60000	1
1	Immobilier	3	Yourthe	10000	1
2	Services	4	Nettoyage	0	2
2	Services	5	Repassage	0	2
4	Cinéma	8	Brad Pitt	1000	4
4	Cinéma	9	Mel Gibson	500	4
4	Cinéma	10	Jean Paul Belmondo	10	4

Autre exemple d'explication : http://www.epershand.net/developpement/mysql-bdd/comprendre-jointures-inner-left-right-join-mysql

16.1.11 Exercices - jointures internes/externes/requêtes jointures/multiples/avec regroupement/champ calculée

Ajoutez la table Pilotes, modèles et villes à l'exercice concernant les avions (point 5.1) :

MODELES

ModID	ModNom	ModVitesse
1	Boeing 747	12000km/h
2	A300	700km/h
3	A310	915km/h
4	Boeing 707	984km/h
5	Concorde	2145km/h
6	Mercure	932km/h

VILLES

VilID	VilNom	VilNbHabitants	VilSuperficie
1	Nice	343304	71,92
2	Paris	2244000	105,4
3	Lyon	484344	47,95
4	Toulouse	441802	118,3
5	Lens	4277	46,61
6	Nantes	284970	65,19

AVIONS

AviID	AviModID	AviNombreDePlaces	AviVilID
1	2	300	1
2	3	300	1
3	5	250	2
4	2	280	3
5	5	160	1
6	1	460	2
7	4	250	2
8	3	300	4
9	6	180	3
10	5	160	2

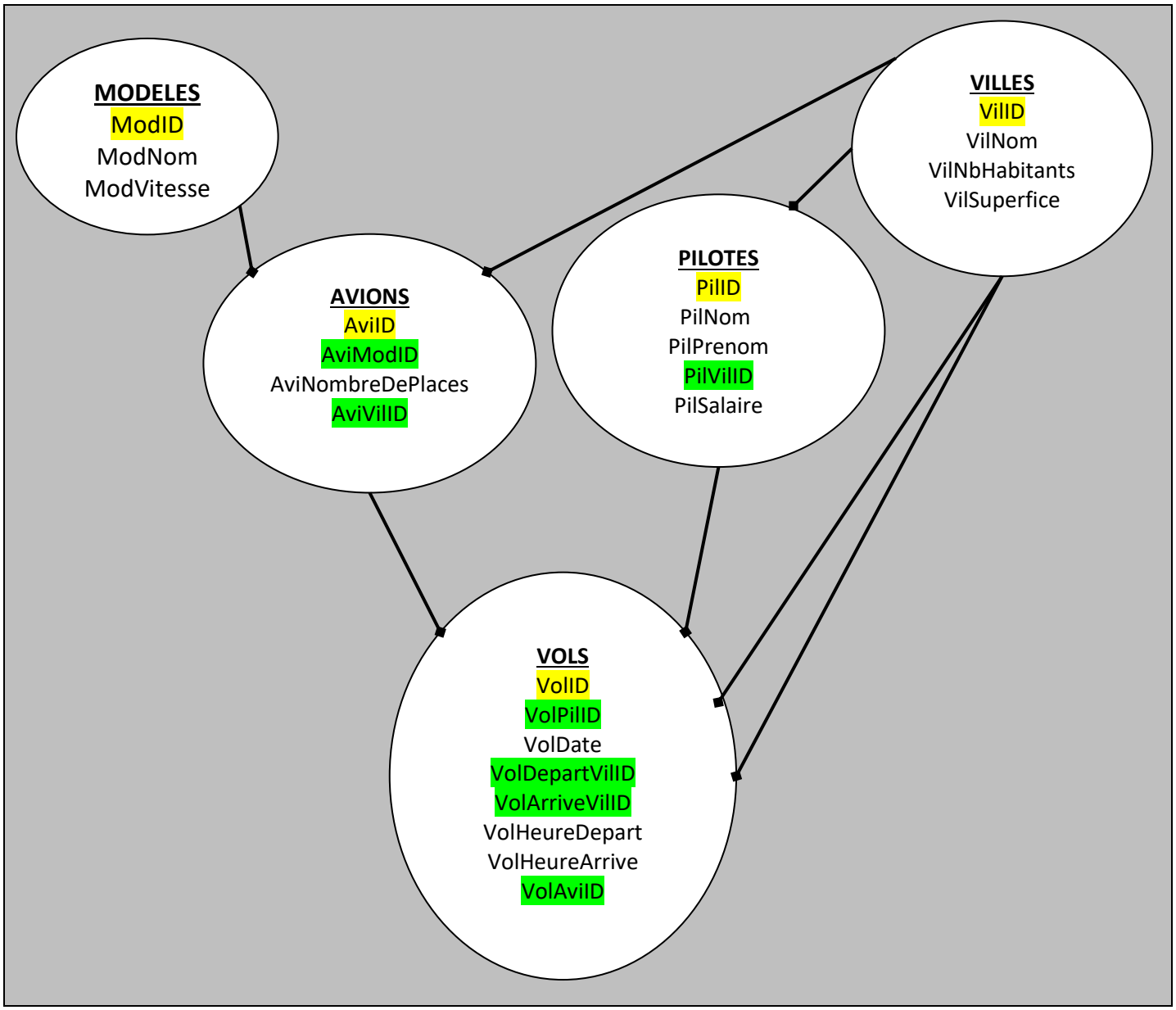
PILOTES

PilID	PilNom	PilPrenom	PilVilID	PilSalaire
1	TIM	Vincent	2	26000.00
2	CLETTE	Lara	4	21000.00
3	AIPAN	Ahmed	1	18000.00
4	TERIEUR	Alain	2	17000.00
5	LAIBOUL	Ella	4	19000.00
6	TERIEUR	Alex	2	18000.00
7	DON	Guy	1	17000.00
8	RATAMAIR	Waldi	3	15000.00
9	OUIN	Serge	5	18000.00
10	GRAFFE	Otto	1	

VOLS

VolID	VolPilID	VolDate	VolDepartVilID	VolArriveVilID	VolHeureDepart	VolHeurearrive	VolAviID
1	1	**2013-10-30**	1	4	11 :00	12 :30	1
2	1	**2013-10-30**	2	4	17 :00	18 :30	8
3	2	**2013-10-30**	4	3	14 :00	16 :00	1
4	5	**2013-10-30**	4	3	18 :00	20 :00	3
5	9	**2013-10-30**	2	1	06 :45	08 :15	1
6	10	**2013-10-30**	3	1	11 :00	12 :00	2
7	1	**2013-10-30**	2	3	08 :00	09 :00	4
8	8	**2013-10-30**	1	2	07 :15	08 :45	4
9	1	**2013-10-31**	6	3	09 :00	15 :30	8
10	8	**2013-10-31**	1	2	12 :15	13 :45	2
11	9	**2013-10-31**	2	3	15 :00	16 :00	2
12	1	**2013-10-31**	3	6	16 :30	20 :00	2
13	4	**2013-10-31**	1	5	11 :00	14 :00	5
14	3	**2013-10-31**	5	2	15 :00	16 :00	5
15	8	**2013-10-31**	2	4	17 :00	18 :00	9
16	7	**2013-10-31**	2	4	18 :00	19 :00	5

Schéma des tables en relation



1. Donner toutes les informations sur les pilotes
2. Donner le nom et le prénom des pilotes
3. Sélectionner l'identificateur et le nom de la ville de chaque ville
4. Sélectionner les noms des pilotes gagnant plus de 25000 €
5. Quels sont les noms des pilotes gagnant entre 20000 et 25000 € ?
6. Quel est la vitesse des boeings?
7. Quels sont les noms des pilotes dont le salaire est inconnu?
8. Quelles sont les villes de départ des différents vols
9. Sélectionner les noms des pilotes habitant Paris
10. Quelles sont les capacités des avions de type Airbus ?
11. Quels sont les vols au départ de Nice desservant Paris
12. Quels sont les avions (identifiant de l'avion + nom du modèles) de capacité supérieure à 250 personnes ou localisés à Paris ?
13. Quels sont les vols au départ de Paris et dont l'heure d'arrivée est inférieure ou égale à 15h00 ?
14. Quel est le salaire moyen des pilotes parisiens ?
15. Trouver le nombre de vols au départ de Paris.
16. Trouver le nom des pilotes effectuant des vols au départ de Paris ?
17. Trouver le nom des pilotes effectuant des vols au départ de Paris sur des Airbus.

18. Quels sont les avions localisés dans la même ville que l'avion numéro 3.
19. Quels sont les pilotes dont le salaire est plus élevé que le salaire moyen des pilotes ?
20. Quels sont les noms des pilotes niçois qui gagnent plus que tous les pilotes parisiens ?
21. Donner le nom des pilotes niçois qui gagnent plus qu'au moins un pilote parisien.
22. Rechercher les pilotes ayant même ville et même salaire que TIM.
23. Donner la liste des pilotes parisiens par ordre de salaire décroissant puis par ordre alphabétique des noms.
24. Quel est le nombre de vols effectués par chacun des pilotes ?
25. Trouver le nombre de vols par pilote, en affichant à chaque fois le modèle et de numéro d'avion.
26. Donner le nombre de vols par pilote seulement s'il est supérieur à 5.
27. Donner le nom des pilotes effectuant au moins 5 vols.
28. Quels sont les numéros d'avions qui ne volent pas ?

16.2 Requêtes ensemblistes

Soit deux tables de même schéma :

Client

Nom	Chiffre
Belgacom	1150
Dupont	19950
Electrabel	3750
Informatifacile	3560
Format Logique	15750
New Form	16840

Fournisseur

Nom	Chiffre
ABComputer	28500
Belgacom	6250
Carglass	1125
Electrabel	9520
Informatifacile	3560
Transport Claude	12500

16.2.1 Union

L'opérateur UNION produit un résultat qui possède les attributs des relations d'origine et tous les tuples de chacune.
Pour obtenir l'union des relations Client et Fournisseur :

```
SELECT * FROM Client
UNION
SELECT * FROM Fournisseur;
```

Nom	Chiffre
Belgacom	1150
Dupont	19950
Electrabel	3750
Informatifacile	3560
Format Logique	15750
New Form	16840
ABComputer	28500
Belgacom	6250
Carglass	1125
Electrabel	9520
Informatifacile	3560
Transport Claude	12500

Pour n'obtenir que les noms des relations **`Client`** et **`Fournisseur`**.

```
SELECT NOM FROM Client
UNION
SELECT NOM FROM Fournisseur;
```

16.2.2 Intersection

L'intersection[5] produit un résultat qui possède les tuples identiques de chacune sur base d'un ou plusieurs attributs spécifiés.

```
SELECT Nom FROM Client
WHERE Nom IN (SELECT Nom FROM Fournisseur);
```

Ou

```
SELECT Nom FROM Client
INNER JOIN FOURNISSEUR
USING (Nom) ;
```

[5] La commande SQL INTERSECT n'existe pas en MySQL !

Pour obtenir les noms des tiers qui sont à la fois client et fournisseur avec leur chiffre d'affaires côté fournisseur.

Client

Nom	***Chiffre***
Belgacom	**1150**
Dupont	**19950**
Electrabel	**3750**
Informatifacile	**3560**
Format Logique	**15750**
New Form	**16840**

Fournisseur

Nom	*Chiffre*
ABComputer	28500
Belgacom	6250
Carglass	1125
Electrabel	9520
Informatifacile	3560
Transport Claude	12500

Nom	
Belgacom	6250
Electrabel	9520
Informatifacile	3560

```
SELECT Nom, Chiffre FROM Fournisseur
WHERE Nom IN (SELECT Nom FROM Client);
```

ou

```
SELECT Nom, Chiffre FROM Fournisseur
INNER JOIN Client
USING (Nom)
```

16.2.3 Différence

Client

Nom	***Chiffre***
Belgacom	**1150**
Dupont	**19950**
Electrabel	**3750**
Informatifacile	**3560**
Format Logique	**15750**
New Form	**16840**

Fournisseur

Nom	*Chiffre*
ABComputer	28500
Belgacom	6250
Carglass	1125
Electrabel	9520
Informatifacile	3560
Transport Claude	12500

Nom
Dupont
Format Logique
New Form

La différence produit un résultat qui possède les tuples de la première table qui, sur base d'un ou plusieurs attributs spécifiés, n'appartiennent pas à la deuxième.

```
SELECT Nom FROM Client
WHERE Nom NOT IN (SELECT Nom FROM Fournisseur);
```

Pour obtenir les noms et chiffres d'affaires des tiers qui sont uniquement fournisseurs.

```
SELECT Nom, Chiffre FROM Fournisseur
WHERE Nom NOT IN (SELECT Nom FROM Client);
```

ou (*forme utile avec un SGBDR qui ne supporte ni l'**EXCEPT**, ni les **SELECT** imbriqués*)

```
SELECT Nom, Chiffre
FROM Client LEFT OUTER JOIN Fournisseur ON Client.Nom = Fournisseur.Nom
WHERE Fournisseur.Nom IS NULL;
```

16.2.4 Le produit cartésien

Le produit cartésien produit un résultat qui possède les tuples de la première table, chacun associé à l'ensemble des tuples de la deuxième table.

Exemple sur une base de données des personnes :

Personne	
IdPers	***Prenom***
1	**Pierre**
2	**Marc**
3	Michel

Communication	
IdCom	*LibCom*
1	Tél.privé
2	GSM

Le produit cartésien donnera

IdPers	*Prenom*	*IdCom*	*LibCom*
1	**Pierre**	**1**	Tél.privé
1	**Pierre**	**2**	GSM
2	**Marc**	**1**	Tél.privé
2	**Marc**	**2**	GSM
3	**Michel**	**1**	Tél.privé
3	**Michel**	**2**	GSM

```
SELECT * FROM Personne, Communication;
```

Bien sûr le résultat du produit cartésien de cet exemple ne présente pas d'informations très utiles. Le suivant permet d'établir la liste des étudiants susceptibles de se présenter à chaque examen organisé.

Etudiant	
IdEtudiant	***Nom***
1	**Panzani**
2	**Ergoton**

Examen
LibExamen
Analyse
Langage C
VB.Net

```
SELECT * FROM Examen, Etudiant;
```

LibExamen	*IdEtudiant*	*Nom*
Analyse	**1**	Panzani
Analyse	**2**	Ergoton
Langage C	**1**	Panzani
Langage C	**2**	Ergoton
VB.Net	**1**	Panzani
VB.Net	2	Ergoton

16.2.5 Exercices -Requêtes ensemblistes

Reprenez la base de données Vols/Avions/Pilotes et effectuez les requêtes suivantes :

1. Sélectionner les avions dont le modèle est A310, A320 ou A330
2. Quels sont les avions dont le modèle est différent de A310, A320 et A330?
3. Quels sont les numéros des pilotes pilotant les avions 2 et 4 ?
4. Quels sont les numéros des pilotes pilotant les avions 2 ou 4 ? (en utilisant une requête ensembliste !)
5. Quels sont les numéros des pilotes pilotant l'avion 2 mais jamais le 4 ?
6. Quels sont les numéros des pilotes qui pilotent tous les avions de la compagnie ?

16.2.6 Exercices récapitulatifs

16.2.6.1 Bibliothèque

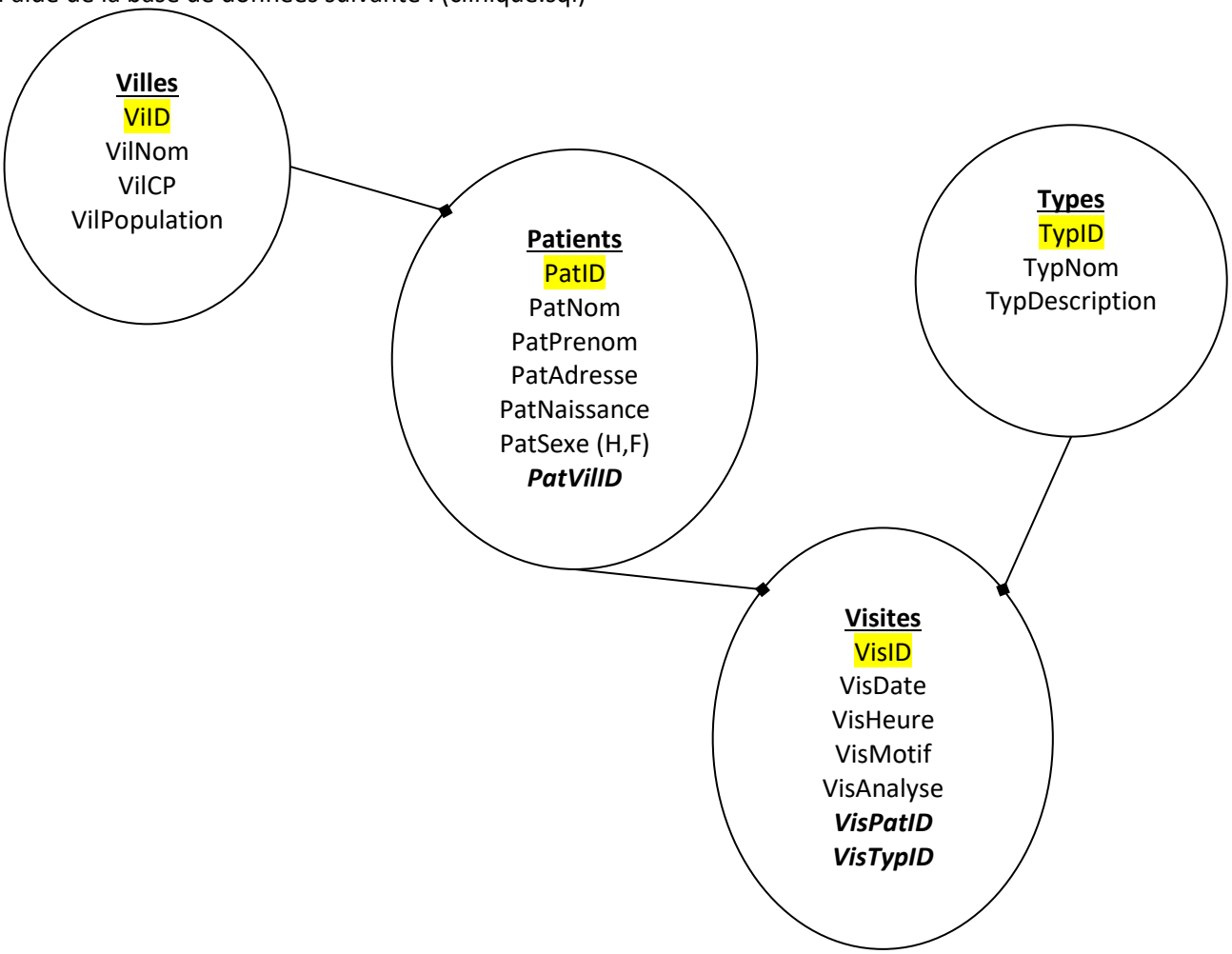
À l'aide de la DB « bibliotheque.sql »

1. En face de chaque titre d'ouvrage (par ordre alphabétique) afficher le nom et prénom de son auteur.
2. Afficher le nom et le prenom des emprunteurs suivi de la date de ses emprunts.
3. Affichez le titre et l'auteur des ouvrages empruntés suivi du nom de leur emprunteurs.
4. A la suite de problèmes de saisie informatique, il existe un certain nombre d'emprunteurs sans emprunts. Affichez les nom d'emprunteurs, suivi de l'identifiant de tous les emprunts qu'il y en ait un ou non
5. Quels sont les personnes qui n'ont jamais emprunter de livre ?
6. Réunissez dans une seule liste tous les titres de livres contenant le mot « nuit » et tous les titres contenant le mot « noire » par ordre décroissant de titre.

7. On veut afficher le titre des livres qui contiennent à la fois « noire » et « nuit ».
8. Affichez dans une colonne les noms et dans un autre colonne les téléphones des personnes (qu'elles aient emprunté ou non) et des editeurs.
9. Sélectionnez sur une colonne Lecteurs/Auteurs tous les emprunteurs qui ont le même nom qu'un auteur.
10. Quels sont les livres qui n'ont jamais été empruntés ?
11. Quels sont les livres qui ont été rendus dans les 15 jours

16.2.6.2 Clinique

À l'aide de la base de données suivante : (clinique.sql)

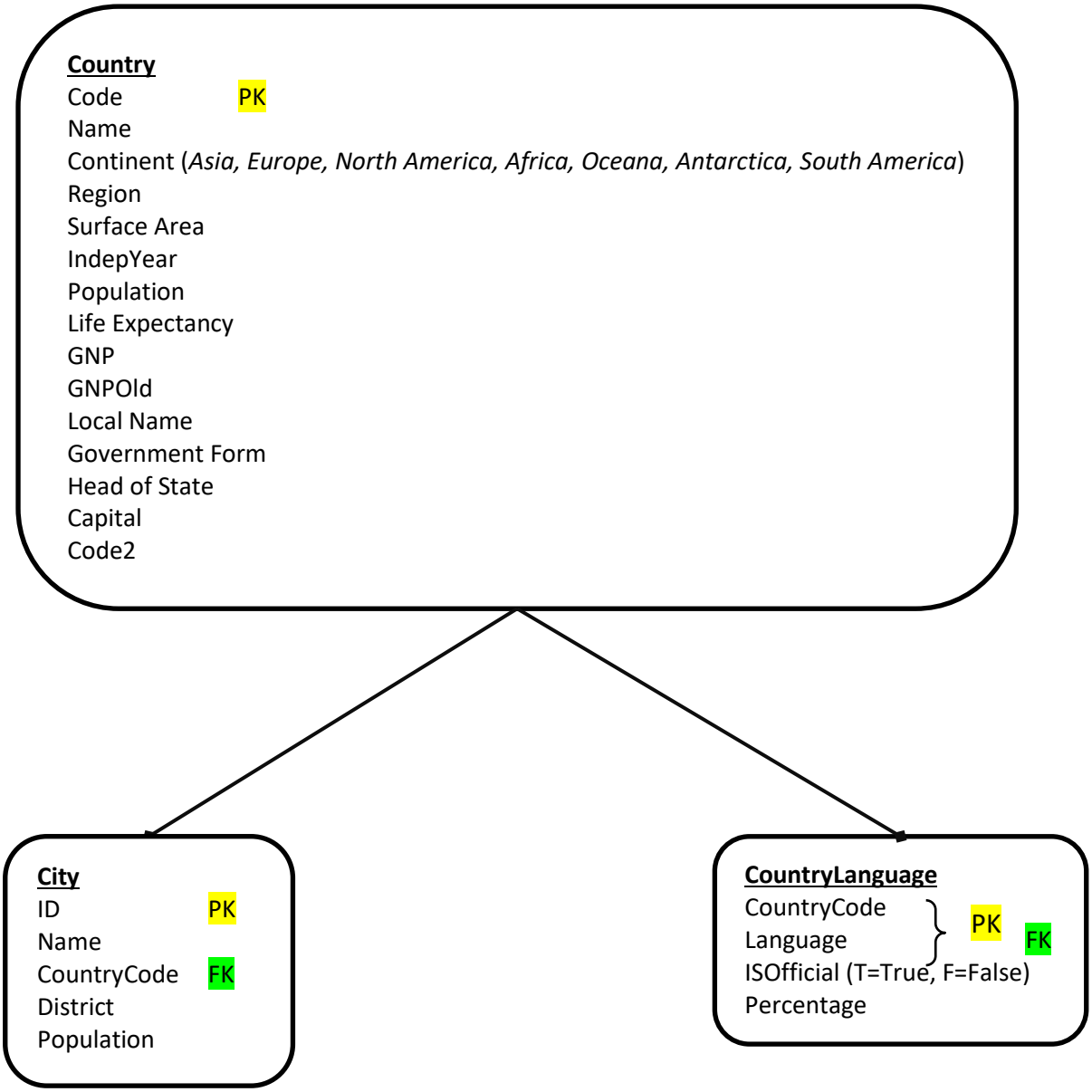


Réalisez les requêtes suivantes :

1. Affichez la liste des patients (nom et prénom), qui n'ont jamais eu de grippe
2. Affichez, pour chaque sexe (Homme, Femme) le nombre de visite qui ont été effectuée avant midi.
3. Affichez la listes des patients (nom et prénom), avec toutes les maladies et description de celle-ci qu'ils ont déjà eu.
4. Quelle est la ville qui a la population la plus élévée ?
5. Affichez la liste des nom et prénom de patients de la ville de Wavre par ordre inversément alphabétique de nom
6. Affichez le nom et prénom des patients qui n'ont jamais été consulter. (ceux qui n'ont pas encore effectués de visite)
7. Affichez la liste des nom et prénom de patients qui sont nés avant 2000 par ordre alphabétique de nom

16.2.6.3 World

1) CREATE DATABASE world ;
2) Insertion et création des tables : « world_innodb.sql »
3) Schéma :



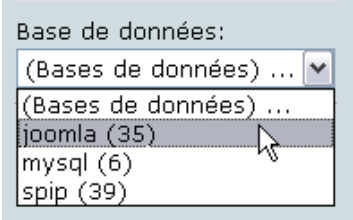
Requêtes SQL

1. Affichez le nom des villes dont la population est inférieure ou égale à 50000 habitants.
2. Quelles sont les langues « Officielles » et leur pourcentage, qui sont parlées en Belgique (Belgium)
3. Combien de langues officielles et combien de langues non officielles sont parlées en Belgique ?
4. Quelle est la ville qui a la population la plus élevé ?
5. Affichez les « district » des Etats-Unis (USA) par ordre inversement alphabétique, dont la population totale du district dépasse les 3000000 habitants.
6. Affichez le nom des villes d'Europe ou on parle français dans le pays.

17. Importation/Exportation des données

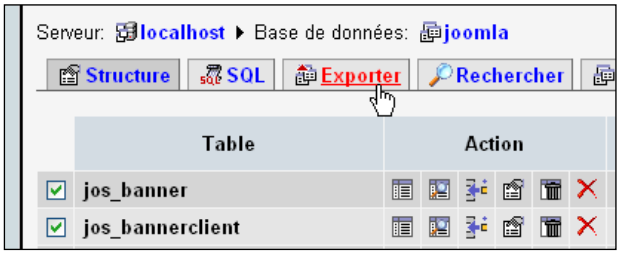
17.1 Réalisation d'un backup sous PHPMyAdmin

a. Localhost/phpmyadmin
b. Sélectionnez la base de données par exemple "Joomla" :



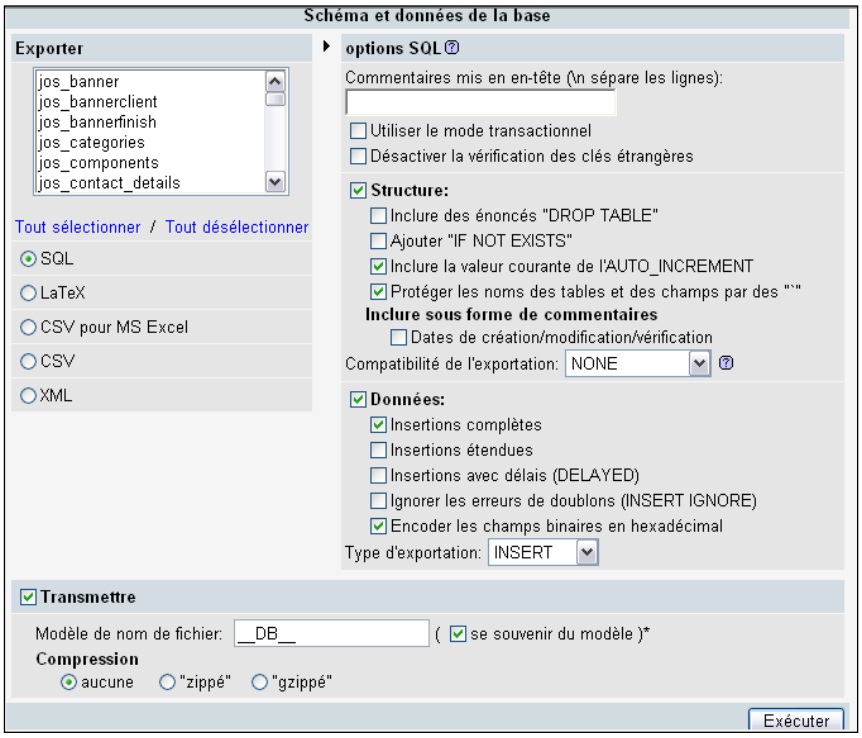
c. Sélectionnez toutes les tables :

d. Utilisez la fonction "Exporter" :



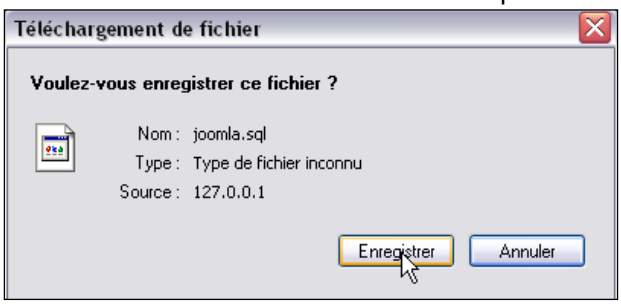
e. Sélectionnez les options :
   - "Inclure la valeur courante de l'AUTO_INCREMENT"
   - Protéger les noms des tables
   - Insertions complètes

- Transmettre



f. Et appuyez sur "Exécuter"

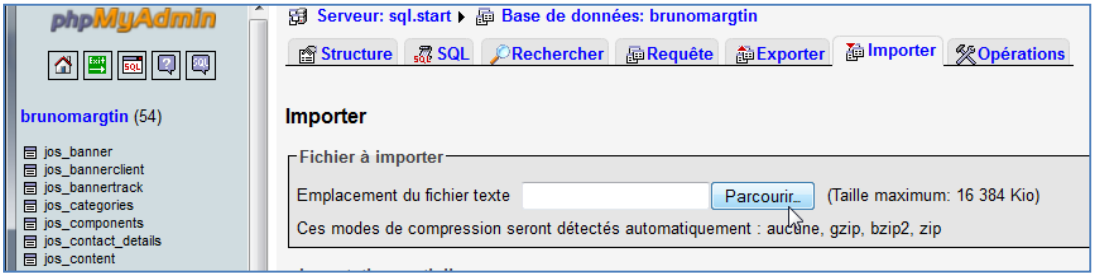
g. Donnez la destination où le fichier sera copié.



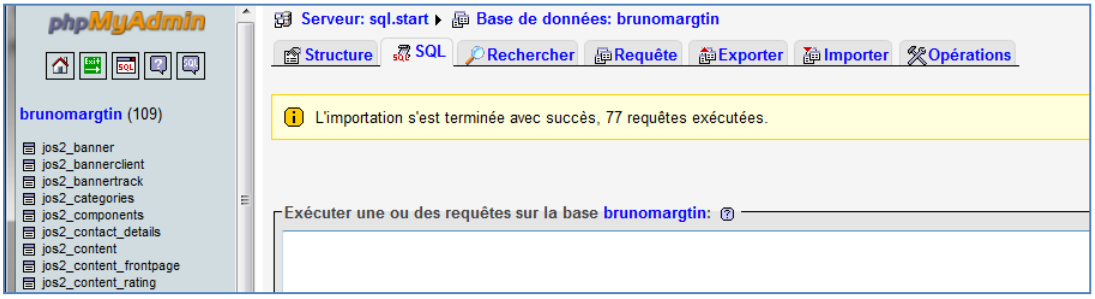
17.2 Restaurer une base de donnée sous PHPMyAdmin

a. Sélectionnez ou créez une base de données.

b. Cliquez sur "Importer" puis "parcourir" pour aller rechercher votre fichier ".sql"

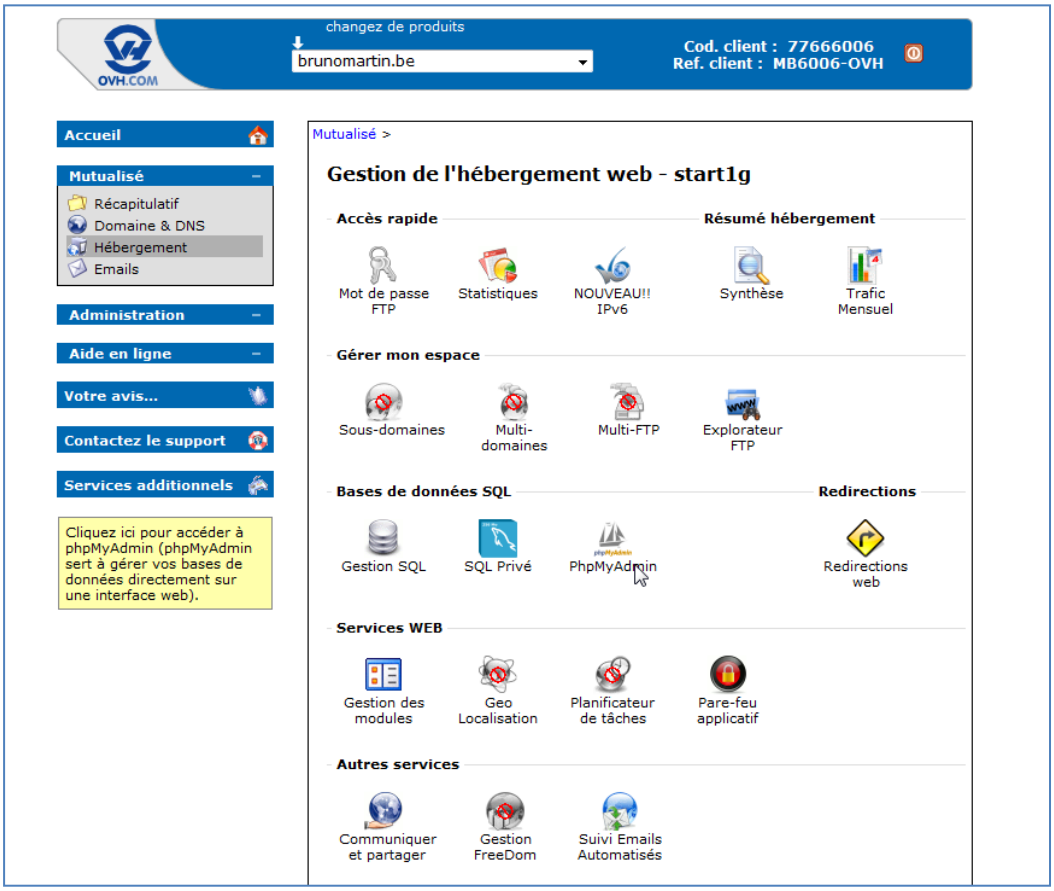


c. Les nouvelles tables apparaissent dans la base de données sélectionnée :

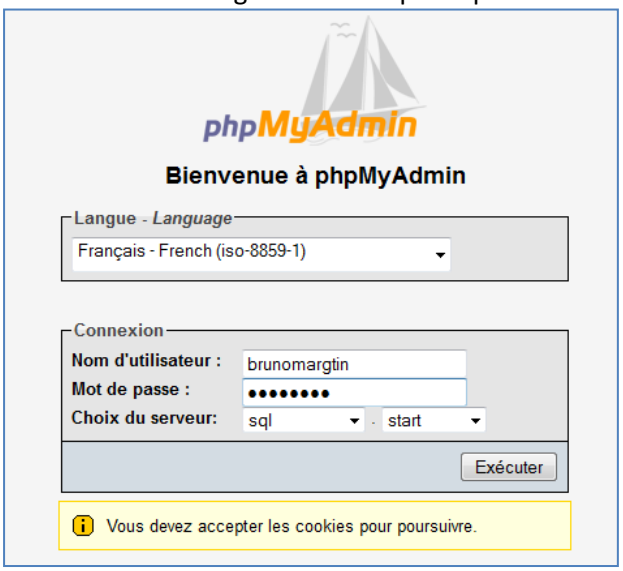


17.2.1 Exemple d'importation sous OVH

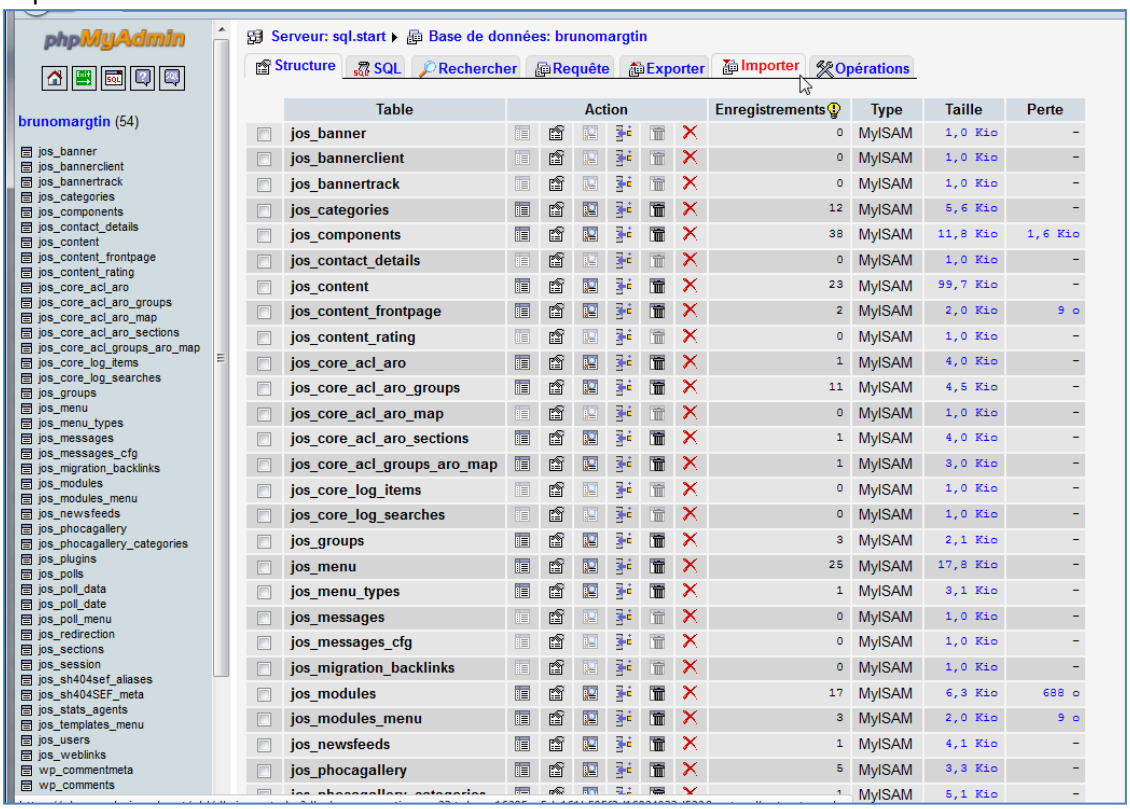
a. Sous ovh, connectez-vous à votre manager à l'aide de votre login et mot de passe, sous la rubrique "Hébergement", cliquez sur "phpmyadmin" :



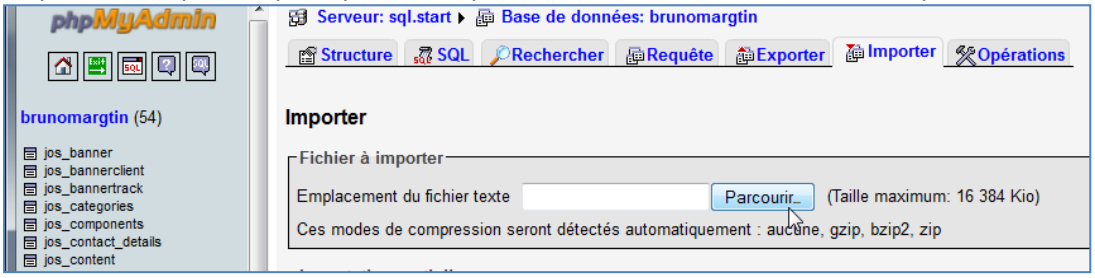
b. Introduisez votre login et mot de passe pour accéder à votre base de données



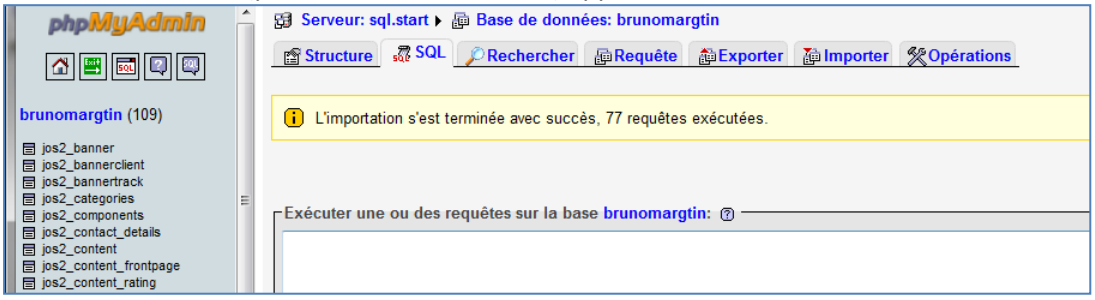
c. Importer vos tables dans la base de données existante :



d. Cliquez sur "Importer" puis "parcourir" pour aller rechercher votre fichier ".sql"



e. Les nouvelles tables (préfixées différemment !) apparaissent :



17.3 Exporter une base de données MySQL

"[6]MySQL propose plusieurs façon d'exporter des données. La principale est la commande en ligne *mysql* permettant de réaliser à peu près n'importe quelle action sur les bases de données qu'elle contient à partir d'une simple ligne de commande :

```
mysql -h host -u user -p password base_de_donnees > fichier_dump
```

La notation suivante est également possible :

```
mysql --host host --user user --pass password base_de_donnees >  fichier_dump
```

[6] Source http://www.commentcamarche.net/contents/mysql/mysqlimport.php3

- host représente le nom ou l'adresse IP de la machine sur laquelle la base de données que vous désirez exporter est installée. Par défaut il s'agit de *localhost*, c'est-à-dire la machine à partir de laquelle la commande *mysql* est lancée
- user représente l'utilisateur avec lequel vous désirez vous connecter. Par défaut il s'agit de l'utilisateur *root*
- password représente le mot de passe de l'utilisateur avec lequel vous désirez vous connecter. Si vous n'indiquez pas de mot de passe, celui-ci sera demandé de manière interactive. Il ne doit pas y avoir d'espace entre -p et le mot de passe fourni, contrairement aux autres champs
- base_de_donnees est le nom de la base de données à exporter.
- fichier_dump est le nom du fichier dans lequel la base va être exportée. Si aucun chemin absolu n'est précisé, le fichier sera stocké dans le même répertoire que la commande mysql. Attention de ne pas lui donner un nom d'un fichier existant dans ce répertoire !"

18. Interface avec PHP

18. 1 Choix de l'API

"[7]Il est recommandé d'utiliser soit l'extension mysqli, soit l'extension PDO_MySQL. Il n'est pas recommandé d'utiliser l'ancienne extension mysql pour de nouveaux développements sachant qu'elle est obsolète depuis PHP 5.5.0, et sera supprimée dans un futur proche.»

18. 2 PDO

PDO est une **classe PHP** destinée à permettre à PHP de communiquer avec un serveur de données. (PDO : Php Data Object)
PDO est ce qu'on appelle une **couche d'abstraction**, c'est à dire qu'il va permet de communiquer avec n'importe quel server de base de données : MySQL, Oracle, Postgresql, etc... (En tout cas sur des requêtes simples).
PDO va permettre (c'est son intérêt majeur) de **sécuriser** les requêtes et de favoriser la réutilisation du code grâce aux **requêtes préparées**.

PDO (**P**HP **D**ata **O**bjects) vise à permettre la création d'un code comportant des accès aux BDD en faisant abstraction du moteur de SGBD utilisé. A partir de PHP 6, c'est le système par défaut activé pour se connecter à une base de données, il est donc nécessaire de commencer à réécrire une partie de ses applications dans ce sens afin de préparer l'avenir…

Avantages de PDO :

- Interface pour SGBD : plus besoin de s'occuper de savoir quelle SGBD est derrière (en théorie) ;
- Orienté objet : les objets PDO et PDOStatement peuvent être étendus, il est donc tout à fait possible de personnaliser et remodeler une partie du comportement initial ;
- Exception : les objets de l'interface PDO utilisent des exceptions, il est donc tout à fait possible d'intégrer facilement un système de gestion des erreurs.

18. 2.1 Activation - Php Windows

Avec PHP 5, cette extension n'est pas activée par défaut. Ouvrez le 'php.ini' et décommentez la ligne suivante :

```
extension=php_pdo.dll
```

Et activez le driver correspondant au type de BDD que vous souhaitez utiliser.
Pour MySQL :

```
extension=php_pdo_mysql.dll
```

Pour Oracle :

```
extension=php_pdo_odbc.dll
```

Vous pouvez trouver le tableau de correspondance des DLL avec le SGBD sur
http://fr.php.net/manual/fr/pdo.drivers.php

[7] http://www.php.net/manual/fr/mysqlinfo.api.choosing.php

18.2.1 Connexion :

Création d'une connexion

```
<?php
// Connexion au serveur
$dns = 'mysql:host=localhost;dbname=toto';
$utilisateur = 'root';
$motDePasse = '';
$connexion = new PDO( $dns, $utilisateur, $motDePasse );
?>
```

Quand le **port** est celui utilisé par défaut par le moteur de base de données, le spécifier est facultatif. Dans notre cas par exemple, le moteur de base de données est MySQL, par conséquent, préciser le port 3606 est inutile (c'est le port par défaut).

```
// DNS où le port est spécifié
$dns = 'mysql:host=localhost;dbname=toto;port=3606';
```

18.2.2 Gestion d'erreur

PDO étant orienté objet, il lève des exceptions en cas de problème. Les erreurs levée sont des **PDOException**, voici les différents types d'erreurs que vous pouvez rencontrer lorsque vous instanciez un objet PDO.

18.2.2.1 Erreur "could not find driver"

Cette erreur survient si vous avez **mal renseigné le moteur de base de données** dans le DNS ou si le driver choisi n'est pas supporté par votre serveur. Dans le cas du driver MySQL, il est généralement actif par défaut sur la majorité des hébergements (activé par défaut sur Wamp).

18.2.2.2 Erreur "Unknown MySQL server host"

Cette erreur surviens quand le nom du serveur est mal renseigné / indisponible (MySQL dans notre cas).

18.2.2.3 Erreur "Can't connect to MySQL server"

Dans le cadres des accès distants (le serveur MySQL n'est pas sur la même machine que le serveur Web), Vous obtiendrez ce message si le **serveur est planté / indisponible**, ou si le serveur auquel vous tenter d'accéder n'est pas un serveur de données MySQL.
Le message d'erreur appraît généralement après un certain temps (assez long) de chargement.

18.2.2.4 Erreur "Unknown database <db_name>"

Le **nom de la base de données est incorrect**, ou plus grave, la base de données n'existe plus...

18.2.2.5 Erreur "Access denied for user"

Vous n'avez pas le droit d'accéder au serveur. Soit votre identifiant, soit votre mot de passe, soit les deux sont mal renseignés.

18.2.2.5 Intercepter les erreurs de connexion avec un try { catch }

Le bloc try / catch est très pratique pour intercepter ce type d'erreur (on appelle ce genre d'erreur des **Exception**) :

```
// Connexion au serveur
try {
  $dns = 'mysql:host=localhost;dbname=toto';
  $utilisateur = 'root';
  $motDePasse = '';
  $connection = new PDO( $dns, $utilisateur, $motDePasse );
} catch ( Exception $e ) {
  echo "Connection à MySQL impossible : ", $e->getMessage();
  die();
}
```

La fonction die() porte bien son nom, elle va tuer le script PHP (toutes les lignes situées après le die ne seront jamais exécutées).

Il est fortement recommandé de se créer un fichier php (par exemple "connexion.php") contenant ces quelques lignes. En effet, si l'une de vos pages nécessite alors un accès à la base de données, il vous suffira d'inclure ce fichier avec require_once.

```
require_once('connexion.php');
```

18.2.3 Méthodes : exec et query

PDO fait la distinction entre deux familles de requêtes :

- QUERY - Pour récupérer une information (SELECT)
- EXEC - Pour un apporter des changements (INSERT, UPDATE, DELETE)

18.2.3.1 SELECT - QUERY

PDO nous offre la liberté d'utiliser la réponse au format que nous voulons, nous pouvons dire que nous voulons traiter les enregistrements reçus comme des tableaux, comme des objets typés, etc...

Pour **sélectionner des enregistrements**, nous utiliserons la méthode query($requete).

```
// On établit la connexion
require_once('connexion.php');

// On envoie la requête
$select = $connection->query("SELECT * FROM t_contact");
```

La variable $select contient maintenant le résultat de la requête, mais sous une forme un peu particulière : un **PDOStatement**[8].
Cet objet contient la réponse du serveur de données à la requête que nous lui avons envoyé. Cet objet va nous permettre de gérer l'affichage des données reçues :

18.2.3.2 setFetchMode / fetch[9]

PDOStatement::fetch — Récupère la ligne suivante d'un jeu de résultats PDO

PDO::FETCH_OBJ : retourne un objet anonyme avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats

Notre variables **`$select`** contient maintenant un objet pour chaque enregistrement obtenu, pour traiter tous les résultats nous allons utiliser la boucle while :

```
// Nous traitons les résultats en boucle et c'est lors de l'utilisation de fetch() que nous spécifions le format de
// récupération pour le traitement.
while( $enregistrement = $select->fetch(PDO::FETCH_OBJ) )
{
  // Affichage d'un des champs
  echo '<h1>', $enregistrement->Nom, ' ', $enregistrement->Prenom, '</h1>';
}
```

PDO::FETCH_OBJ : retourne un objet anonyme avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats

Tester un résultat d'une requête vide :

```
if (!$select->fetch()) {
   echo "requete vide";
} else {
        //traitement par boucle
}
```

[8] http://www.php.net/manual/fr/class.pdostatement.php : Documentation officiel pour PDOStatement
[9] http://php.net/manual/fr/pdostatement.fetch.php

18.2.3.3 UPDATE et DELETE - EXEC

```
// Ajout de données
try {
  $connection->exec("INSERT INTO t_contact VALUES('','Mr','Térieur','Alex','Rue des Gnous','1','1300','Wavre',
'0479/784583','alexterieur@yahoo.fr')");
    } catch ( Exception $e ) {
  echo "Une erreur est survenue lors de l'insertion ";
}
```

11.2.4 Les requêtes préparées

Le principe est simple, vous n'allez pas exécuter directement les requêtes, vous allez définir un **gabarit de requête** avec des "options".

Avantages de cette méthode :

- optimisation des performances pour des requêtes appelées plusieurs fois ;
- protection des injections SQL (plus besoin de le faire manuellement) ;

18.2.4.1 Paramètres mystères

Utilisation du caractère ? pour transmettre des valeurs au moment de l'execution :

```
// Préparation de la requête
$selectionPrepa = $connection->prepare('SELECT * FROM t_contact WHERE id=? LIMIT 1');
try {
  // On envoi la requête
  $selectionPrepa->execute(array(1));

  // Traitement
  if( $enregistrement = $selectionPrepa->fetch(PDO::FETCH_OBJ)){
    echo '<h1>', $enregistrement->Nom, ' ', $enregistrement->Prenom,  '</h1>';
  }
} catch( Exception $e ){
  echo 'Erreur : ', $e->getMessage();
}
```

Ce système permet de **réutiliser** la requête préparée par la suite en modifiant la valeur utilisée lors de l'execution par une autre.

18.2.4.2 Paramètres nommés

```
$selectionPrepa = $connection->prepare('SELECT * FROM t_contact WHERE nom LIKE :search OR prenom LIKE
:search');
try {
  // On envois la requète
  $selectionPrepa->execute(array('search'=>'%le%'));

  // Traitement
  while( $enregistrement = $selectionPrepa->fetch(PDO::FETCH_OBJ)){
    echo '<h1>', $enregistrement->Nom, ' ', $enregistrement->Prenom,  '</h1>';
  }

} catch( Exception $e ){
  echo 'Erreur : ', $e->getMessage();
}
```

```
$insert = $connection->prepare('INSERT INTO t_contact(Titre,Nom,Prenom,Adresse,Numero,CP,Ville,GSM,EMail)
VALUES(:titre, :nom, :prenom, :adresse, :numero, :cp, :ville, :gsm, :email)');
try { $success = $insert->execute(array(
'titre'=>'Mr',
```

```
'nom'=>'Dus',
'prenom'=>'Jean-Claude',
'adresse'=>'Rue des bronzes',
'numero'=>'8',
'cp'=>'1300',
'ville'=>'Wavre',
'gsm'=>'0799/464646',
'email'=>'jcdussss@gmail.com'));
  if( $success ) {
    echo "Enregistrement réussi";
  }
} catch( Exception $e ){
  echo 'Erreur de requète : ', $e->getMessage();
}
```

18.2.4.3 Sécuriser sans passer par des requêtes préparées

Si on passe par des requêtes non préparées, pour que la requête soit sécurisée, il existe la méthode "quote". Cette méthode s'utilise directement avec la ressource de connexion.

```
<?php
$RessourceDeConnection = new PDO(...); // ouverture d'une connexion
$RessourceDeConnection->query("SELECT id_membre FROM membres WHERE nom =
".$RessourceDeConnection ->quote($nom, PDO::PARAM_STR));
?>
```

Nous ouvrons une nouvelle connexion, ensuite nous faisons une requête de type SELECT avec la méthode query prévue à cet effet.

Remarquez que pour appeler le nom, nous utilisons directement la ressource de connexion à laquelle nous ajoutons la méthode quote, en premier argument on met la valeur qui devra être protégée et en deuxième argument (qui est optionnel), on met le type de valeur à protéger.

Valeurs possibles pour le deuxième argument :

- **PDO::PARAM_STR** : pour une chaîne de caractères ; (=valeur par défaut)
- **PDO::PARAM_INT** : pour le type 'integer' de SQL ;
- **PDO::PARAM_NULL** : pour le type NULL de SQL ;
- **PDO::PARAM_BOOL** : pour un booléen ;
- **PDO::PARAM_LOB** : pour le type 'objet large' de SQL.

Attention, lorsque vous utilisez cette méthode, n'englobez aucune valeur avec '' (pour une chaîne de caractère), la méthode se chargera de le faire pour vous si c'est nécessaire.

Ressources :

http://fmaz.developpez.com/tutoriels/php/comprendre-pdo/

http://fr.php.net/manual/fr/book.pdo.php

18.3 Exercices 9 - PHP/ Base de données

9.1 Pilotes – « listepilotestableau.php »

À l'aide de la base de données « ryanair.sql »

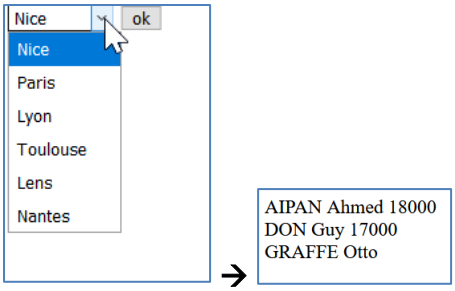
Affichez la liste des pilotes dans un tableau :

TIM	Vincent	26000
CLETTE	Lara	21000
AIPAN	Ahmed	18000
TERIEUR	Alain	17000
LAIBOUL	Ella	19000
TERIEUR	Alex	18000
DON	Guy	17000
RATAMAIR	Waldi	15000
OUIN	Serge	18000
GRAFFE	Otto	

Affichez la liste des pilotes ainsi que leurs villes : « **listepilotesvilles.php** »

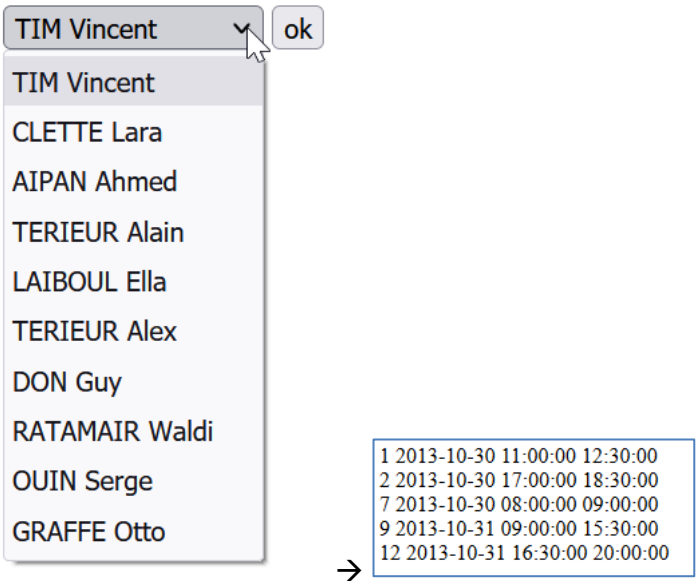
TIM Vincent 26000 Paris
CLETTE Lara 21000 Toulouse
AIPAN Ahmed 18000 Nice
TERIEUR Alain 17000 Paris
LAIBOUL Ella 19000 Toulouse
TERIEUR Alex 18000 Paris
DON Guy 17000 Nice
RATAMAIR Waldi 15000 Lyon
OUIN Serge 18000 Lens
GRAFFE Otto Nice

Dans un formulaire, créez une liste déroulante avec les villes **« listevilles.php »** et affichez les pilotes correspondant **« afficherpilotes.php »**



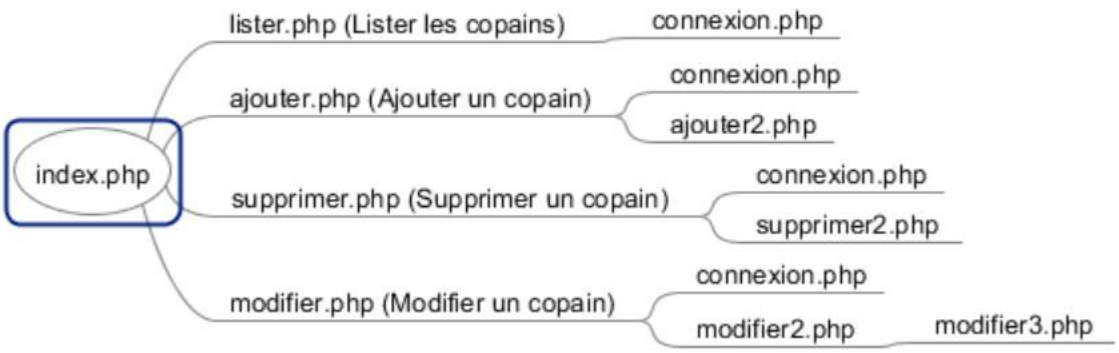
AIPAN Ahmed 18000
DON Guy 17000
GRAFFE Otto

Dans un formulaire, créez une liste déroulante avec les pilotes **« listepilotes.php »** et affichez les vols correspondant **« listevols.php »**



9.2 Copains

Créez une base de données « **amis** » contenant une table « **copains** » avec un « nom » et « numéro de téléphone » et réalisez les pages suivantes :



9.3 Deconinck

Reprenez le site réaliser au cours de XSTA (Deconick) et faites en sorte que l'entièreté du site soit dynamique.

19. Sources

La référence PHP (anglais & français) :
http://www.php.net
La référence MySQL (anglais) :
http://www.mysql.com
Des cours en ligne :
http://www.developpez.com
L'outil phpMyAdmin :
http://phpmyadmin.sourceforge.net
Un aperçu de la nouvelle sauvegarde sous MySQL 6.0 :
http://cedric-duprez.developpez.com/tutoriels/mysql/sauvegarde6
Le langage de requête SQL :
http://sql.developpez.com/
Implémentez vos bases de données relationnelles avec SQL (openclassrooms) :
https://openclassrooms.com/fr/courses/6971126-implementez-vos-bases-de-donnees-relationnelles-avec-sql
MySQL 8.0 Reference Manual :
https://dev.mysql.com/doc/refman/8.0/en/

20. Evaluation

Pour atteindre le seuil de réussite, l'étudiant sera capable :

face à une structure informatique opérationnelle connectée à Internet, disposant des logiciels appropriés et de la documentation nécessaire, en utilisant le vocabulaire technique et l'orthographe adéquate, en respectant les normes et standards en vigueur,

et au départ d'un cahier des charges contenant un projet de pages web dynamiques en interaction avec une source de données externe (système de base de données, XML,…) :

- de générer un ensemble de pages web contenant un système de navigation et un contenu dynamiques intégrant formulaires et résultats ;
- d'établir et de construire une structure de la base de données, compatible de la situation présentée dans le cahier des charges, sur un système de gestion de bases de données relationnelles en produisant les contenus nécessaires ;
- de sélectionner, d'ajouter, de supprimer et de modifier les données sur la source externe.

Pour la détermination du degré de maîtrise, il sera tenu compte des critères suivants :

- le respect des consignes figurant dans le cahier des charges,
- la pertinence des commentaires dans le code,
- la lisibilité du code,
- la pertinence des choix et des techniques,
- le degré d'autonomie atteint.