

WEB DESIGNER UI/UX

Approche développement backend

5XBAK-1

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 les applications dynamique et leurs besoins en termes de scripts serveurs et de bases de données ;
- ◆ de décrire et de caractériser une base de données ;
- ◆ d'identifier 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'interagir avec une source de données externe pour des requêtes de sélection (simple, multiple, avec tri, avec filtre ...),
- ◆ d'identifier différents langages utilisés pour la programmation côté serveur ;
- ◆ d'analyser un script serveur en termes de:
 - définition des variables,
 - structures conditionnelles et itératives,
 - fonctions,
 - classes, propriétés et méthodes
 - ...
- ◆ d'exploiter, dans un langage choisi, les variables, les structures conditionnelles, les structures itératives, les tableaux, l'affichage dans une application web dynamique,... ;
- ◆ de transférer des données vers le serveur au moyen de méthodes telles que GET, POST,... en veillant au respect des bonnes pratiques en matière de transfert de données;
- ◆ de mettre en œuvre une application web dynamique au moyen d'un framework ;
- ◆ de recourir à bon escient à la documentation disponible.

Table des matières

1. Introduction	3
1.1 Préparer son ordinateur	4
2. Premiers pas avec PHP	5
2.1 L'instruction echo	5
2.2 Inclure des portions de page	5
Exercice 1 : include	7
3. Les variables	7
3.1 Types de données	7
3.1.1 String (chaîne de caractères)	7
3.1.2 int (nombre entier)	7
3.1.3 Le type float (nombre décimal)	8
3.1.4 Le type bool (booléen)	8
3.1.5 Les tableaux	8
3.2 Afficher le contenu d'une variable	9
3.3 Les opérations de base : addition, soustraction	9
Exercice 2 : les variables	10
4. Structures conditionnelles	10
4.1 if	10
4.2 Switch	11
5. Les répétitives	12
5.1 While	12
5.2 for	12
5.3 La boucle foreach	13
Exercice 4 : répétitive	13
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	18
7.2 Passage de variables dans l'URL : la méthode GET	20
Exercice 6 : Les formulaires	20
8. Gestion des bases de données – Théorie des bases de données relationnelles	23
8.1 Définitions	23
9. Administration avec l'outil phpMyAdmin	24
9.1 Présentation	24
9.2 Pourquoi utiliser la ligne de commande ?	25
10. Création d'une base de données	25
10.1 Identification et création de tables	25
10.2 Identification des champs et leurs types de données	26
10.2.1 Types des attributs	26
10.2.2 Exemples de types d'attribut :	26
10.3 Création des relations entre les tables	26

10.3.1 Clé primaire	26
Exercice 7 – création de table	27
11. Gestion du contenu de la base de données	27
11.1 Ajout de données	27
11.2 Modification de données	28
11.3 Suppression de données	29
Exercice 8 – insertion de données	29
12. Intégrité référentielle	29
Exercice 9 – Analyse/ Intégrité référentielle	31
13 Interrogation de base de données	31
13.1 Requêtes de sélection	31
13.1.1 Simple.....	32
13.1.2 Avec filtre	33
13.1.3 Avec tri	33
Exercice 10 – Sélection simple/filtre/tri	34
13.1.4 Multiple.....	34
13.1.5 Les jointures internes	35
13.1.6 Les jointures externes	36
Regroupement	37
13.1.7 Champs calculés	38
13.1.8 Fonctions de MySQL	39
Exercice 11 - jointures internes/externes/requêtes jointures/multiples/avec regroupement/champ calculée	40
14. Interface avec PHP	44
14.1 PDO	44
14.1.1 Connexion :	44
14.1.2 Méthodes : query	45
14.1.3 INSERT, UPDATE et DELETE	46
Exercice 13 - PHP/ Base de données.....	46
14. Sources	48
Javascript ou php coté serveur ? - GPT-3.5 – Août 2023	48
15. Evaluation	49

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, d'interagir avec une base de données, etc.

>> Il génère une page HTML.

1.1 Préparer son ordinateur

Il existe des paquetages (Wamp, Xampp, Easyphp, ...) tout prêts pour Windows pour travailler avec un **serveur local**.

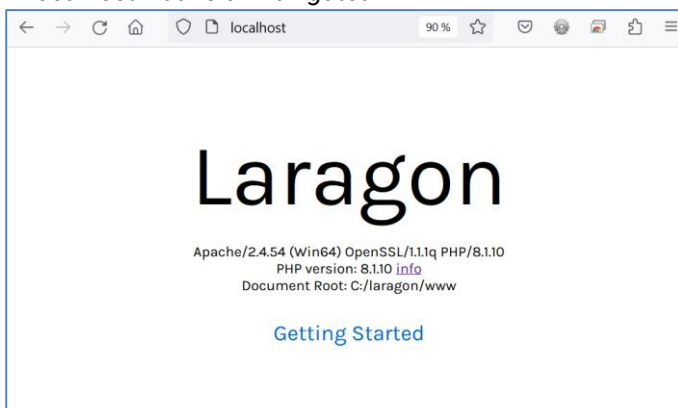
Dans le cadre de ce cours, nous utiliserons Laragon :

- **Configuration simplifiée** : Laragon vise à simplifier la configuration de l'environnement de développement local en fournissant une installation rapide et facile de PHP, Apache, MySQL, etc.
- **Interface conviviale** : Laragon propose une interface utilisateur graphique intuitive qui permet de gérer les services et les projets de manière simple. Cela peut être particulièrement utile si vous préférez une approche visuelle.
- **Prise en charge de différents CMS et frameworks** : Laragon prend en charge divers CMS populaires comme WordPress, Joomla, Drupal et des frameworks tels que Laravel, Symfony, etc., avec des fonctionnalités intégrées pour les installer rapidement.
- **Facilité de gestion des bases de données** : Laragon offre des fonctionnalités pour gérer les bases de données, y compris l'importation, l'exportation et la gestion des utilisateurs.
- **Isolation des environnements** : Laragon propose une fonctionnalité appelée "Isolators" qui permet de créer des environnements isolés pour les projets, ce qui signifie que vous pouvez utiliser différentes versions de PHP ou d'autres logiciels pour différents projets.

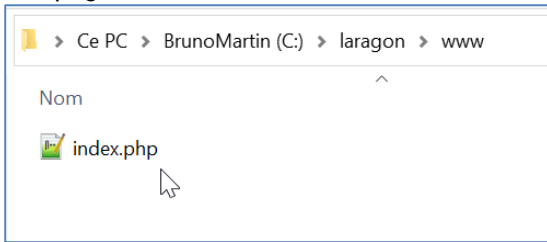
Laragon contient 3 choses :

- **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. Il permet d'enregistrer des données de manière organisée (comme la gestion du contenu d'un site, la liste des membres d'un site, etc.).

Une fois le programme installé et démarré, vous pouvez alors lancer la page d'accueil de Laragon en tapant « **Localhost** » dans un navigateur :



Vos pages se trouveront dans le dossier « C:\Laragon\www »



La création d'une « arborescence » bien organisée pour vos projets web sera nécessaire afin de faciliter la gestion et la navigation dans vos applications web.

2. Premiers pas avec PHP

Il est conseillé d'utiliser un éditeur de texte qui colore le code source (comme « Notepad++ ») pour programmer convenablement en 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 /* */

[illegible]

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

```
<?php
//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.

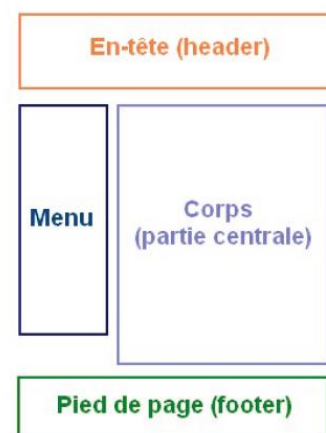
```
<?php
    echo " Bonjour ";
?>
```

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

Enregistrer le fichier avec l'extension .php

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 guillemets :

```
<?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.

>> 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 où 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

>> 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

>> 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 affiche la même chose en utilisant les variables (\$bonjour, \$qui, \$quoi) – exe2.2.php
- 2.3 Écrire une page qui affiche dans un tableau html (<table>) – exe2.3.php

```
bonjour1 bonjour2
bonjour3 bonjour4
bonjour5 bonjour6
```
- 2.4 Écrire une page php qui affiche la même chose en utilisant des variables – exe2.4.php
- 2.5 Écrire une page php qui affiche la même chose en utilisant un tableau – exe2.5.php
- 2.6 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) – exe2.6.php

4. Structures conditionnelles

Une structure conditionnelle permet d'exécuter ou non une série d'instructions en fonction d'une condition d'origine (appelée aussi expression ou prédicat). Si le calcul de cette condition retourne TRUE alors le bloc d'instructions concerné est exécuté.

4.1 if

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

```
<?php
$age = 8;
if ($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 />";
}
?>
```

```
<?php
$age = 8;
if ($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...

```
<?php
$nombre=5;
if ($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)

```
<?php
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;
}
?>
```

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

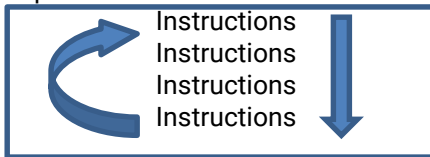
Exemple :

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 :

```
<?php
$chiffre= 1;
while ($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'**incrément**ation 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'écho 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 en HTML) – exe4.5.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

- 4.6 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

Example :

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 « 0 » de 10 sur 10

[illegible]

- 4.14 Affichez un triangle de « O »

```
O
OO
OOO
OOOO
OOOOO
OOOOOO
OOOOOOO
OOOOOOOO
OOOOOOOOO
OOOOOOOOOO
```

- 4.15 Affichez un triangle de « O »

```
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOO
OOOOOOOO
OOOOOOO
OOOOOO
OOOOO
OOOO
OOO
OO
O
```

- 4.16 Affichez un triangle de « O »

```

O
  OO
    OOO
      OOOO
        OOOOO
          OOOOOO
            OOOOOOO
              OOOOOOOO
                OOOOOOOOO
                  OOOOOOOOOO
```

- 4.17 Affichez un triangle de « O »

```

OOOOOOOOOO
  OOOOOOOOO
    OOOOOOOO
      OOOOOOO
        OOOOOO
          OOOOO
            OOOO
              OOO
                OO
                  O
```

- 4.18. 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 X O O O X O
O O O O X O X O
```

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

```
0 0 0 0 0 0 0 0 0
0 0 0 0 0 X 0 0 X 0
```

- 4.19 Affichez la forme suivante

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

- 4.20 Affichez un carré de 10x10 en « * », mais uniquement avec une diagonale.

```
0
0
0
0
0
0
0
0
0
0
```

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.

```
<?php
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>

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";
}
echo div(5,0)."<br>"; //utilisation de la fonction
```

```
?>
```

```
<?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 séparé et le nommer lib.inc (=bibliothèque)

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

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 transitent 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.

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"]`² (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];
?>
```

² on peut écrire indistinctement : `$_POST['prenom']`, `$_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>
```

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

Votre adresse e-mail

☒ abonnement gratuit à notre-planete.info

Description de votre site

Type d'échange souhaité :

Dans quelle rubrique souhaitez-vous être présent ?

☐ Environnement / Ecologie

☐ Géographie

☒ Photos

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 **htmlspecialchars()** 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 ``. 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**

Quelle est votre moyenne?

1
2
3
4
5
6
7
8
9
10
11

0≤moyenne <10 : refusé
 10≤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**

Nom :

Prénom :

Adresse :

Numéro :

Boîte :

Ville :

CP :

Email :

Tel :

Fax :

Gsm :

Pays : Belgique ▼

Femme? ☐

Homme? ☐

Autre? ☐

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**):

Brut Hors Taxe :

Type:

Grossiste
Détaillant
Particulier

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.

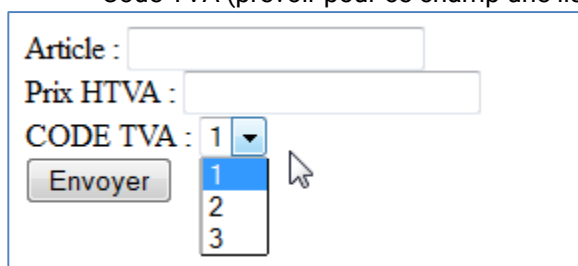
- 6.7 Scouts - Ecrivez un programme qui en fonction d'une variable informera l'enfant de sa catégorie **exe6.7.php – resultat6.7.php**

- 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

- 6.9 **exe6.8.php – resultat6.8.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 :

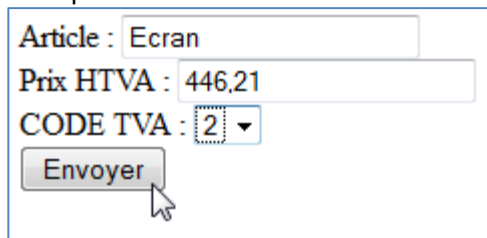
CODE TVA1 = 2,10%

CODE TVA2 = 5,50%

CODE TVA3 = 19,60%

Le nom de l'article + son prix TVAC

Exemple :



Affichera à l'écran :

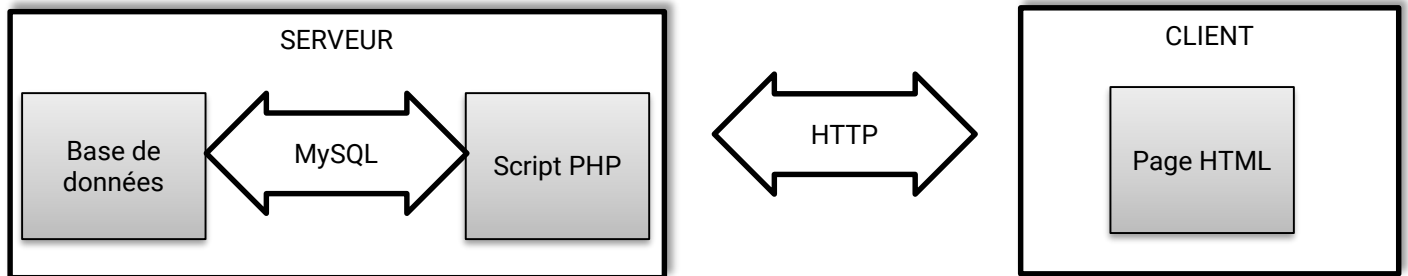
Ecran PRIX TVAC = 470.75155

8. 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.



8.1 Définitions

Relation : C'est en fait un tableau à deux dimensions 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*

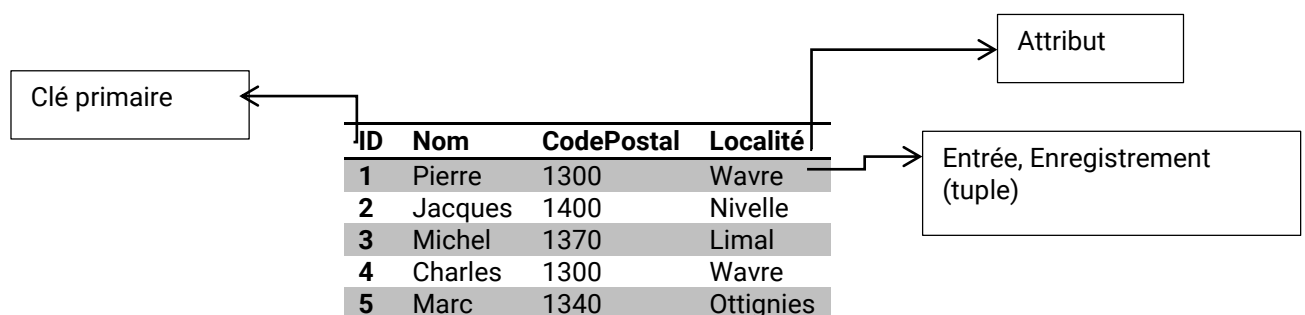
Personnes			
Nom	Prenom	Adresse	telephone
Dupond	Marc	8, Rue de l'octet	0123456789

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

Enregistrement: liste des valeurs d'une ligne d'une relation.

Une relation est un peu une classe (programmation orientée objet) qui ne posséderait 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.



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

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

CP	Nom
1300	Wavre
1400	Nivelle
1370	Limal
1340	Ottignies

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".

9. Administration avec l'outil phpMyAdmin

9.1 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 disponibles via cette interface graphique, toutefois, phpMyAdmin vous montrera toujours la commande SQL qui sera effectuée :

```

✓ Votre requête SQL a été exécutée avec succès

ALTER TABLE `uforum` DROP PRIMARY KEY ,
ADD PRIMARY KEY ( `id` , `Number` ) ;

```


9.2 Pourquoi utiliser la ligne de commande ?

- ✘ Les interfaces graphiques permettent de faire pas mal de choses, mais une fois que vous vous mettez 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*)

10. 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 ;
```

10.1 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 'chaîne1']
  Colonne2 Type2 [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaîne2']
  Colonne3 Type3 [NOT NULL | NULL] [DEFAULT valeur3] [COMMENT 'chaîne3']
  ...
[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
- ✘ NomContrainte1 typeContrainte1 : 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

10.2 Identification des champs et leurs types de données

10.2.1 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 VARCHAR(40) !

10.2.2 Exemples de types d'attribut :

Entiers _____	INT
Nombres réels (flottants) _____	FLOAT
Chaîne de caractère de taille variable (M -> 1 à 255 caractères) _____	VARCHAR(M)
Texte de 65535 caractères max _____	TEXT
Texte de 4294967295 caractères max _____	LONGTEXT
Date au format anglophone AAAA-MM-JJ _____	DATE
Heure au format HH:MM:SS _____	TIME
Énumération de chaîne de valeur _____	ENUM('valeur','valeur2',...)

10.3 Création des relations entre les tables

10.3.1 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...

Num_reg_nat INT(11) PRIMARY KEY

Exercice 7 – 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 ?

11. Gestion du contenu de la base de données

11.1 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 *warnings*.
- ✘ 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, 0000000000000000 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')
```

11.2 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 *warnings*.
- ✘ 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.

11.3 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

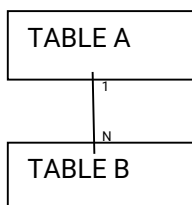
Exercice 8 – insertion de données

Insérez en sql la personne suivante dans la table T_contact créée à l'exercice précédent :

titre	nom	premier	Adresse	Numéro	CP	Ville	gsm	téléphone	email
Dr	ZIEUVAIR	Bruno	Clos de l'Armoise	1323	1300	Wavre	0479/784513	010/212122	zieuvairbruno@yahoo.fr

12. 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.



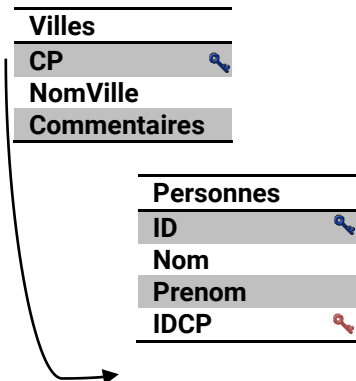
1. Clé primaire : identifiant unique et non nul
2. On ne peut pas ajouter un élément B pour un A inexistant
3. On ne peut pas supprimer un élément A s'il lui correspond un ou plusieurs éléments B

[**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

Exercice 9 – Analyse/ Intégrité référentielle

- ✧ Dessiner le schéma des 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	Louissette	Commercial
5	TLUAUP	Yves	PDG
6	BACH	Gérard	Graphiste
7	DELLILE	Georgette	Graphiste
8	DELAGRANGE	Maurice	Comptable

- ✧ Dessiner le schéma des 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	VolDestinationDepart	VolDestinationArrivé	VolHeureDepart	VolHeureArrivé	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.

13 Interrogation de base de données

13.1 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

13.1.1 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 Soularde	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

13.1.2 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 Soularde	0123456789

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

```
SELECT Nom, Prenom, Adresse, Telephone
FROM personnes
WHERE Nom="Dupond"
```

Nom	Prénom	Adresse	Téléphone
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soularde	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%'
```

13.1.3 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 Soularde	0123456789

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

```
SELECT *
FROM Personnes
ORDER BY prénom ASC
```

Nom	Prénom	Adresse	Téléphone
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soularde	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
```

Exercice 10 – Sélection simple/filtre/tri

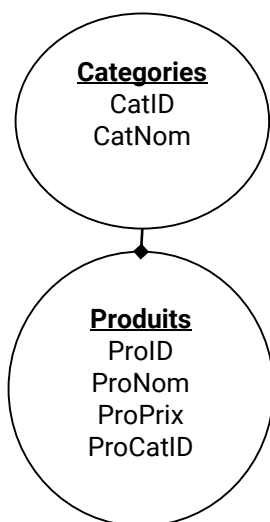
Soit la table "gens"

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":

13.1.4 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

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

SELECT * FROM Categories

CatID	CatNom
1	Immobilier
2	Services
3	Auto
4	Cinéma

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';
```

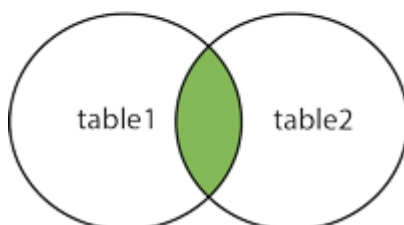
Le résultat de la 1ère requête peut être directement injecté dans la deuxième :

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

L'inconvénient de la méthode précédente est que le SGBD va quand même exécuter deux requêtes. La solution est d'utiliser une jointure interne.

13.1.5 Les jointures internes

INNER JOIN



Les jointures **internes**³ (*inner joins*) ; à chaque ligne de la 1^{er} table est associée toute ligne de la 2^{ème} table vérifiant la condition.

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

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

³ Quand le type de jointure est omis, c'est automatiquement une jointure interne (inner)

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).

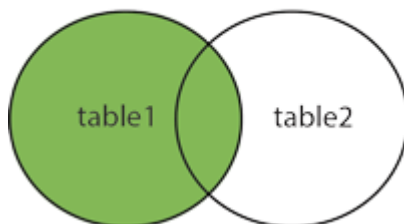
13.1.6 Les jointures externes

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 :

left outer

LEFT JOIN



Une jointure interne est d'abord effectuée, puis y est ajoutée toute ligne de la 1^{ère} table sans concordance avec la 2^{ème} 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

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).

Regroupement sur plusieurs critères :

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

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

13.1.7 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

13.1.8 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 être composés entre eux pour donner des expressions très complexes.

Quelques exemples

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

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.

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%'
```

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.

Exercice 11 - 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

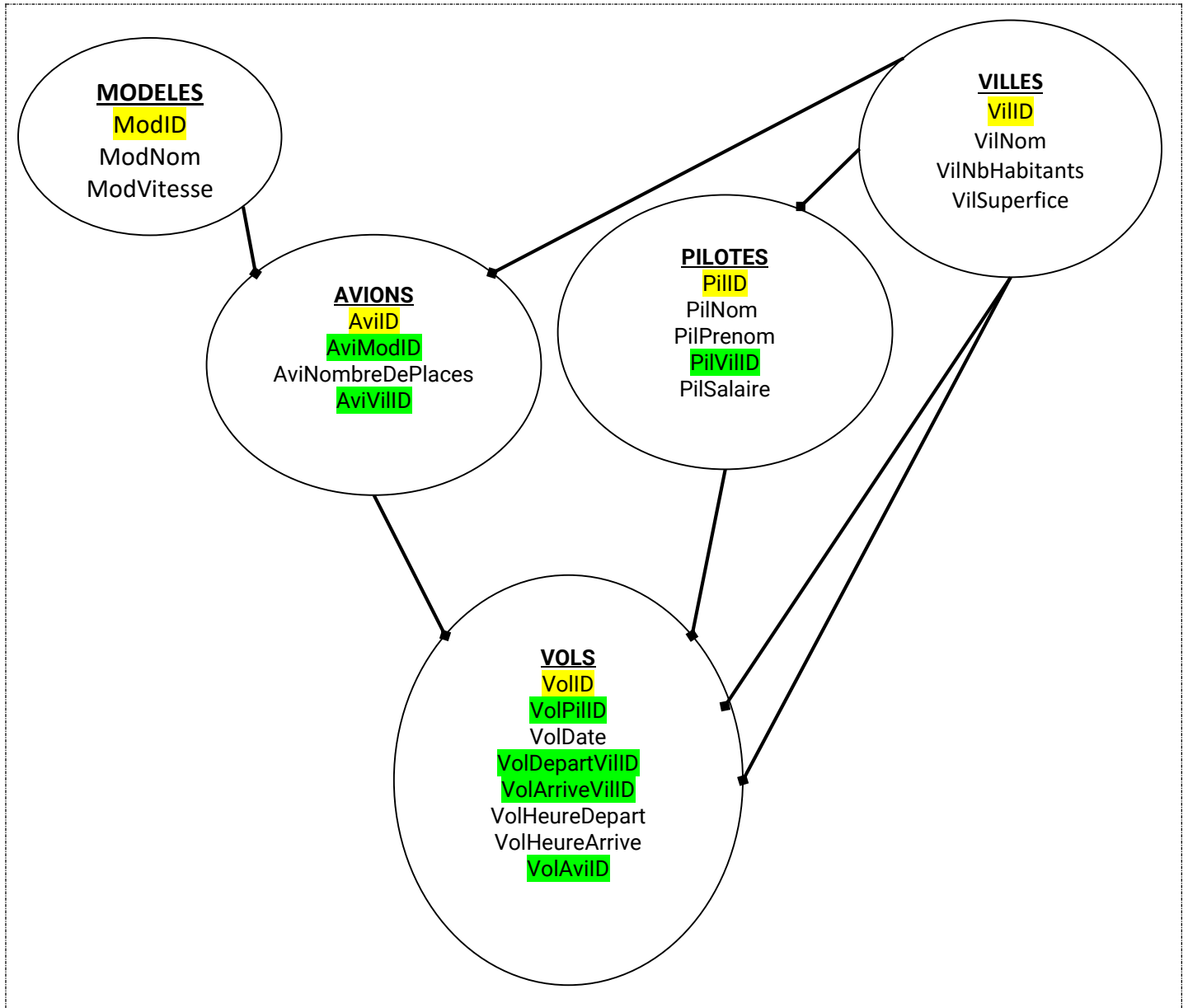
PILOTES

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	OUIIN	Serge	5	18000.00
10	GRAFFE	Otto	1	

VOLS

VolID	VolPilID	VolDate	VolDepartVilID	VolArriveVilID	VolHeureDepart	VolHeurearrive	VolAvilID
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



- Donner toutes les informations sur les pilotes
- Donner le nom et le prénom des pilotes
- Sélectionner l'identificateur et le nom de la ville de chaque ville
- Sélectionner les noms des pilotes gagnant plus de 25000 €
- Quels sont les noms des pilotes gagnant entre 20000 et 25000 € ?
- Quel est la vitesse des Boeings?
- Quels sont les noms des pilotes dont le salaire est inconnu?
- Quelles sont les villes de départ des différents vols
- Sélectionner les noms des pilotes habitant Paris
- Quelles sont les capacités des avions de type Airbus ?
- Quels sont les vols au départ de Nice desservant Paris
- Quels sont les avions (identifiant de l'avion + nom du modèles) de capacité supérieure à 250 personnes ou localisés à Paris ?
- Quels sont les vols au départ de Paris et dont l'heure d'arrivée est inférieure ou égale à 15h00 ?
- Quel est le salaire moyen des pilotes parisiens ?
- Trouver le nombre de vols au départ de Paris.
- Trouver le nom des pilotes effectuant des vols au départ de Paris ?
- Trouver le nom des pilotes effectuant des vols au départ de Paris sur des Airbus.
- Quels sont les avions localisés dans la même ville que l'avion numéro 3.
- 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 ?

14. Interface avec PHP

14.1 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 serveur 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 (**PHP Data Objects**) 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.

mysql vs mysqli

« MySQLi (la lettre « i » vient de « improved » en anglais, qui signifie amélioré) est une extension du langage de programmation PHP, permettant d'accéder aux bases de données MySQL. MySQL fait partie, avec Oracle et Microsoft SQL Server, des systèmes de gestion de base de données relationnelles (SGBD) les plus répandues dans le monde [...] En PHP 7 la fonction **mysql()** n'est plus utilisable, et a été remplacée par **mysqli()**. »⁴

14.1.1 Connexion :

Création d'une connexion

```
<?php
// on se connecte à MySQL et on sélectionne la base
$connection = mysqli_connect('dns', 'utilisateur', 'motdepasse', 'basededonnees',
'port');
?>
```

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 MySQLi, par conséquent, préciser le port 3606 est inutile (c'est le port par défaut).

```
$connection = mysqli_connect('dns', 'utilisateur', 'motdepasse', 'basededonnees', 3606);
```

Intercepter les erreurs de connexion

Pour intercepter les types d'erreur : `connect_errno`:

```
<?php
// vérification de la connexion
if ($connection -> connect_errno) {
    echo "Impossible de se connecter : " . $mysqli -> connect_error;
    exit();
}
?>
```

⁴ <https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/une-introduction-a-mysqli/>

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 la fonction include.

```
<?php
include('connexion.php');
?>
```

14.1.2 Méthodes : query

- ✦ Exécute une requête sur la base de données

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

```
<?php
// On établit la connexion
Include('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 : tableau associatif.

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 :

mysqli_fetch_array

mysqli_fetch_array — Récupère la ligne suivante d'un ensemble de résultats sous forme de tableau associatif.

Retourne une ligne de données de l'ensemble de résultats et la renvoie sous forme de tableau. Chaque appel ultérieur à cette fonction renverra la ligne suivante dans l'ensemble de résultats, ou null s'il n'y a plus de lignes.

En plus d'enregistrer les données sous forme d'un tableau à indices numériques, elle peut aussi les enregistrer dans un tableau associatif, en utilisant les noms des champs comme clés.

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

```
<?php
while( $enregistrement =mysqli_fetch_array($select))
{
    // Affichage des champs
    echo '<h1>' . $enregistrement['Nom'] . ' ' . $enregistrement['Prenom'] . '</h1>';
}
?>
```

Pour vérifier si une requête renvoie des lignes ou non :

```
<?php
// on crée la requête SQL
$req = "SELECT * FROM table;";

// on envoie la requête
$res = $conn->query($req);
// Si on a des lignes...
if ( $res->num_rows > 0 )
    echo "On a des résultats";
} else {
    echo "On n'a aucun résultat";
}
?>
```

14.1.3 INSERT, UPDATE et DELETE

```
<?php
// Ajout de données
// On créé la requête
$req = "INSERT INTO table(texte) VALUES ('Du texte')";

// on envoie la requête
$res = $conn->query($req);
```

```
// Mise à jour de données
// On créé la requête
$req = "UPDATE FROM table WHERE id=XXX";

// on envoie la requête
$res = $conn->query($req);
```

```
// Suppression de données
// On créé la requête
$req = "DELETE FROM table WHERE id=XXX";

// on envoie la requête
$res = $conn->query($req);
```

Exercice 13 - PHP/ Base de données

13.1 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
OUIIN	Serge	18000
GRAFFE	Otto	

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

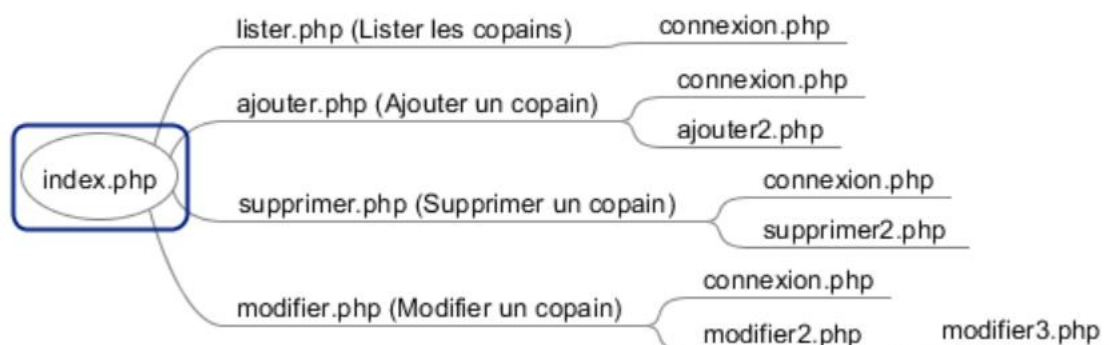
TIM Vincent 26000 Paris
CLETTE Lara 21000 Toulouse
AIPAN Ahmed 18000 Nice
TERIEUR Alain 17000 Paris
LAIBOUL Ella 19000 Toulouse
TERIEUR Alex 18000 Paris
DON Guy 17000 Nice
RATAMAIR Waldi 15000 Lyon
OUIIN Serge 18000 Lens
GRAFFE Otto Nice

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

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

13.2 Copains

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



13.3 Deconinck

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

14. 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 8.1 Reference Manual :

<https://dev.mysql.com/doc/refman/8.1/en/>

<https://www.linuxtricks.fr/wiki/php-passer-de-mysql-a-mysqli-requetes-de-base>

Javascript ou php coté serveur ? - GPT-3.5 – Août 2023

Il convient de noter que le choix entre JavaScript (Node.js) et PHP pour le développement côté serveur dépend largement des besoins spécifiques du projet, des compétences de l'équipe de développement et des considérations techniques. Cependant, voici quelques raisons pour lesquelles le JavaScript côté serveur (Node.js) a gagné en popularité par rapport au PHP dans certains cas :

Un langage de programmation commun : En utilisant JavaScript des deux côtés (frontend et backend), les développeurs peuvent travailler avec un langage cohérent, ce qui peut simplifier le processus de développement et de maintenance.

Asynchronicité et évolutivité : Node.js est conçu pour gérer de manière efficace les opérations asynchrones, ce qui est particulièrement utile pour les applications en temps réel, les applications collaboratives et les applications nécessitant des performances élevées. Cela peut être plus difficile à gérer dans un environnement PHP traditionnel.

Écosystème riche : Node.js bénéficie d'un écosystème de packages (modules) étendu et actif grâce à npm (Node Package Manager). Cela facilite l'intégration de fonctionnalités tierces et accélère le développement.

Partage de code : L'utilisation du même langage de programmation des deux côtés permet de partager certaines parties de la logique de code entre le frontend et le backend, ce qui peut simplifier la maintenance et réduire les erreurs.

Déploiement plus simple : Node.js permet de créer des applications autonomes, ce qui simplifie leur déploiement sur différents serveurs sans avoir à configurer un serveur web séparé comme c'est souvent le cas avec PHP.

Performance : Node.js est conçu pour être performant, en utilisant le moteur JavaScript V8 de Google. Cela

Le choix entre PHP et JavaScript (Node.js) pour le développement côté serveur dépend des besoins spécifiques de votre projet et de vos compétences techniques. Voici quelques raisons pour lesquelles vous pourriez envisager d'utiliser PHP plutôt que JavaScript (Node.js) côté backend :

Historique et adoption : PHP est l'un des langages de programmation côté serveur les plus anciens et les plus largement adoptés pour le développement web. De nombreux sites web populaires et systèmes de gestion de contenu (CMS) sont basés sur PHP, ce qui en fait un choix solide pour les projets qui nécessitent une technologie éprouvée.

Facilité d'apprentissage : PHP est souvent considéré comme plus facile à apprendre pour les débutants en programmation par rapport à JavaScript. Sa syntaxe est plus simple et plus proche du langage naturel, ce qui peut faciliter la transition pour les développeurs novices.

Hébergement partagé : De nombreux fournisseurs d'hébergement web proposent un support natif pour PHP, ce qui rend le déploiement plus simple et plus accessible pour les projets de petite et moyenne taille.

Vaste communauté et documentation : En raison de son ancienneté et de sa popularité, PHP dispose d'une vaste communauté de développeurs et de nombreuses ressources en ligne, ce qui peut être très utile pour résoudre des problèmes et trouver des solutions.

Cadres de développement établis : PHP possède un certain nombre de cadres de développement bien établis comme Laravel, Symfony et CodeIgniter, qui offrent des structures solides pour construire des applications web de manière efficace et organisée.

Intégration avec les bases de données : PHP offre une intégration simple et éprouvée avec divers systèmes de gestion de bases de données, ce qui en fait un choix

peut être particulièrement avantageux pour les applications nécessitant une faible latence et une grande rapidité.

Technologies modernes : Node.js s'intègre bien avec les technologies modernes telles que les applications en temps réel, les API RESTful, les applications monopages (SPA), etc.

Cependant, il est important de noter que PHP reste toujours pertinent pour de nombreux types de projets. Il a une longue histoire, une base d'utilisateurs établie et est particulièrement adapté aux applications web traditionnelles et aux sites dynamiques. Le choix entre JavaScript (Node.js) et PHP dépendra donc des spécificités de votre projet, de vos besoins en matière de performance, de votre équipe de développement et de vos préférences techniques.

populaire pour les applications nécessitant des opérations de base de données.

Compatibilité avec les systèmes existants : Si vous travaillez sur des projets qui nécessitent l'intégration avec des systèmes ou des applications existantes en PHP, il peut être plus judicieux de continuer à utiliser PHP pour la cohérence et la compatibilité.

Performance pour certaines charges de travail : PHP peut être optimisé pour des charges de travail spécifiques, comme les applications basées sur le traitement de formulaires et la manipulation de contenu dynamique.

Cependant, il est important de noter que chaque langage a ses avantages et ses inconvénients, et le choix dépendra toujours des besoins particuliers de votre projet et de l'expertise de votre équipe de développement.

15. 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, éventuellement basé sur un framework, en interaction avec une source de données externe (système de base de données, XML,...) et un template fourni :

- ♦ de générer un ensemble de pages web présentant un contenu dynamique et un système de navigation ;

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

- ♦ le niveau de pertinence technique du code en relation avec les consignes du cahier des charges : l'optimisation du code et la pertinence des commentaires;
- ♦ le niveau de lisibilité du code, (facilité de partage, indentation, conventions et respect des bonnes pratiques d'écriture, ...)
- ♦ le degré d'autonomie atteint dans l'exploitation et l'utilisation de la documentation référencée ;