Initiation à MySQL/MariaDB & au PHP

Michel Scifo

Le Maître Réfleur 2015

Attention

Ce livre numérique ne comporte ni dispositif de cryptage limitant son utilisation ni identification par un tatouage permettant d'assurer sa traçabilité, mais il est sous licence CC-BY-NC-ND (cf Annexe 5) ; à ce titre, il est librement diffusible, à condition de ne pas être vendu seul ou associé.

Nous essayons de respecter le Code de la Typographie, mais quand la tâche s'avère trop complexe ou trop éloignée de nos conceptions, nous passons outre !

Dépôt légal 2e trimestre 2015
ISBN : 978-9537431-5-9

Le Maître Réfleur
12 AV Général Roux
38800 Le Pont-de-Claix

Avertissement

Il s'agit de la version 0.2 du livre Initiation à MySQL & au PHP. Il reste à faire un travail de relecture beaucoup plus important que je ne le pensais, à écrire les cahiers d'exercice, à gérer les annexes, l'index. J'essaierais de finaliser régulièrement une version, afin d'arriver à la version 1 la première semaine de juillet. En l'état, il apporte, déjà, une approche nouvelle de ces deux logiciels.

L'association MySQL/MariaDB-PHP est la base de la majorité des sites dynamiques & des sites interactifs.

Cette formation de deux semaines vise d'abord, la création d'une petite base de données & des tables qu'elle contient grâce à la compréhension des notions élémentaires relatives aux SGBD, illustrées avec MySQL/MariaDB, & enfin, son exploitation avec un navigateur web, au moyen de formulaires créés dans un langage de script tel PHP.

Il faut lever une ambiguïté : les formations de concepteurs développeurs informatiques, à l'AFPA, se font en onze mois, avec un programme plus vaste incluant les méthodes de développement. Nous n'avons que deux semaines pour traiter deux produits riches de fonctionnalités, dont l'étude demanderait six semaines pour chacun, & vous n'avez pas d'expérience du développement. Compte-tenu de l'ampleur des fonctionnalités de ces deux produits & de cette inexpérience, il est illusoire de penser être capable de concevoir, *facilement*, un site web dynamique à l'issue de ces deux semaines. Notre objectif sera plus modeste : faciliter la démarche de conception & celle d'exploitation, en vous montrant :

- une méthode de travail efficace ;
- des rudiments nécessaires à la création d'une base de données & à son exploitation ;
- des bases du langage PHP indispensables pour générer les formulaires facilitant l'exploitation d'une base de données.

Mais même ainsi, il ne serait pas possible d'atteindre ces objectifs, dans le temps imparti, sans que vous acquériez, préalablement les connaissances présentées dans ce document. Cette condition satisfaite, si vous vous mettez au travail, immédiatement après la fin de la seconde semaine, vous arriverez, en y passant du temps certes, à créer & à maintenir un site web dynamique.

Ce cours s'appuie sur la page http://www.siteduzero.com/tutoriel-3-14668-concevez-votre-site-web-avec-php-&-mysql.html?all=1, adaptée à nos besoins. Cette page est sous licence Creative Commons. Le cours complet demande deux mois de travail selon l'auteur.

À un moindre niveau, il s'inspire, également, du site http://www.lephpfacile.com/cours/ dont les droits sont réservés, mais il s'agit d'un bon exemple de cours peu clairs, malgré quelques bonnes idées.

Wikipédia, les manuels en ligne de PHP & de MySQL & de MariaDB ont été abondamment sollicités. D'autres sources & d'autres sites seront employés ponctuellement.

Introduction

Il est possible d'employer MariaDB/MySQL en ligne de commande, avec le client intégré dans le logiciel : *mysql*. Cependant, ce SGBD étant employé essentiellement avec des applications web, c'est-à-dire, avec des applications écrites en php stockées & s'exécutant dans un des dossiers gérés par un serveur web, il s'avère utile de connaître :

- le fonctionnement des pages web contenant les formulaires facilitant l'accès aux bases de données ;
- le PHP puisqu'il sert à écrire la majorité des pages web dynamiques ;
- le client graphique phpMyAdmin (www.phpmyadmin.net).

Qu'est-ce qu'une page web ?

C'est l'affichage dans un navigateur, ou butineur, d'un contenu, présenté d'une certaine façon, après l'exécution de certaines actions :

- la présentation, dans un fichier css, dit feuille de style, généralement commun à toutes les pages du site, afin d'assurer une uniformité de présentation ;
- le contenu est défini dans un fichier html (page statique) ou php (page dynamique) ; il peut être intégré dans le fichier ou accessible dans des fichiers annexes (images, sons, films) au moyen d'une référence à ces fichiers :
- les actions résultent de l'exécution de scripts *javascript* stockés dans le fichier html ou dans un fichier annexe.

L'idéal serait d'avoir un fichier css par site contenant toutes les options de présentation. C'est, en pratique peut réaliste : en effet, quand vous devez faire une présentation unique, la formaliser dans un

fichier css prend plus de temps que de la laisser dans la page html. Ainsi, dans le cas de la première version de notre site personnel, l'entête & le bas de page étaient dans des fichiers inclus dans toutes les pages. Nous n'avions, donc, pas stocké les présentations s'y rapportant dans la feuille de style du site.

QU'EST-CE QU'UN SITE WEB ?

???Un site est un ensemble de fichiers, dit pages *ouebes* ou *webs*, d'images ou d'autres éléments multimédias & de scripts. Il est *statique*, quand toutes les pages sont stockées en HTML sur le serveur ou *dynamique*, quand elles sont générées par des scripts, PHP entre autres, stockés sur le serveur.

Une page web est un fichier texte écrit en langage HTML ou XHTML, le second étant une extension du premier. Un fichier css est un fichier texte de balises html, contenant uniquement des informations relatives à la présentation des informations. Les pages HTML sont interprétés par le navigateur (client) ayant servi à les demander au serveur. Le rendu diffère d'un client à l'autre, car tous les navigateurs ne respectent pas intégralement les préconisations de programmation du W3C, l'organisme qui définit ce langage.

Les fichiers de scripts sont exécutés par le client (JavaScript) ou par le serveur (PHP).

- *Site statique* : réalisé uniquement à l'aide de fichiers en [X]HTML & en CSS 01001. Il fonctionne très bien mais son 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. Ce n'est pas très pratique si on doit le mettre à jour plusieurs fois dans la même journée !

Les sites statiques sont donc bien adaptés pour réaliser des sites *vitrine*, pour présenter par exemple son entreprise, ne nécessitant pas des mises à jour fréquentes. Ce type de site se fait de plus en plus rare aujourd'hui, car dès que l'on rajoute un élément d'interac-

tion (comme un formulaire de contact), on ne parle plus de site statique mais de site dynamique.

* *Site dynamique* : en plus de *(X)HTML* & de *CSS*, il emploie un langage de script (*PHP*) & souvent un SGBD (*MySQL/MariaDB*). Son contenu est *dynamique* parce qu'il peut changer sans l'intervention du *webmestre* ! La plupart des sites web, que vous visitez, sont des sites dynamiques. La seule condition nécessaire pour apprendre à créer ce type de site est de déjà savoir réaliser des sites statiques en *(X)HTML* & en *CSS*. Dans ces cas-là, les fichiers ont une extension indiquant l'interpréteur à employer : *php* pour le *PHP*, *asp* pour l'*ASP*, etc.

Voici quelques éléments que vous serez en mesure de réaliser, à terme :

- *un espace membres* : vos visiteurs pourront s'inscrire à votre site & avoir accès à des sections qui leur sont réservées ;
- *un forum* : un forum de discussion permettant à ses utilisateurs de s'entraider ou de passer le temps ;
- *un compteur de visiteurs* : pour déterminer le nombre de visiteurs qui se sont connectés dans la journée sur votre site, ou le nombre de visiteurs en train de l'utiliser ;
- *des actualités* : pour automatiser l'écriture d'actualités, en offrant à vos visiteurs la possibilité d'en rédiger, de les commenter, etc. ;
- *une newsletter* : pour envoyer un e-mail à tous vos adhérents régulièrement afin de leur présenter les nouveautés & de les inciter ainsi à revenir sur votre site.

Cela sera facilité par la diffusion d'outils logiciels de plus en plus performants comme les outils *CMS* (*Content Management Systems*, *systèmes de gestions de contenu*), tels *Wordpress* ou *Drupal*.

Bien entendu, ce ne sont là que des exemples. Il est possible d'aller encore plus loin, tout dépend de vos besoins. Sachez par exemple que la quasi-totalité des sites de jeux en ligne sont dyna-

miques. On retrouve notamment des sites d'élevage virtuel d'animaux, des jeux de conquête spatiale, etc.

Mais... ne nous emportons pas ! Avant de pouvoir en arriver là, vous avez de la lecture & bien des choses à apprendre ! Commençons par la base : savez-vous ce qui se passe lorsque vous consultez une page web ?

Leur fonctionnement

Lorsque vous voulez visiter un site web, vous tapez son adresse dans votre navigateur web, que ce soit Mozilla Firefox, Internet Explorer, Chromium, Safari ou un autre. Mais ne vous êtes-vous jamais demandé comment faisait la page web pour arriver jusqu'à vous ?

Il faut savoir qu'internet est un réseau composé d'ordinateurs & de routeurs, chaque élément de ce réseau est repéré par une adresse internet, dite *adresse IP*, composée, pour le moment de 4 chiffres chacun compris entre 1 & 254. Certaines adresses sont permanentes, celles des serveurs & des routeurs en particulier, d'autres sont provisoires, celles des clients (vous & nous).

Quand vous voulez vous connecter à un site, vous n'indiquez pas son adresse que vous ne connaissez pas, mais son nom. Il faut que quelque part des ordinateurs serveurs puissent retrouver l'adresse internet correspondant à ce nom, on les appelle des serveurs de noms, ils sont le cœur d'internet.

Avant cela il faut que le routeur le plus proche de votre domicile établisse la route permettant d'accéder aux serveurs de votre fournisseur d'accès, qui eux s'adresseront au serveur de noms.

Les moteurs de recherche sont des serveurs particuliers qui permettent de trouver, thématiquement, des pages internet.

Il y avait en décembre 2014, plus d'un 1 milliards de sites (mais seulement 180 millions étaient actifs) gérés par un peu plus de 5 millions de serveurs (source www.netcraft.com).

Les routeurs étant gérés par les opérateurs de télécommunication, on peut, en simplifiant, dire que l'on trouve sur internet, deux sortes de matériels : les clients & les serveurs.

- *Les clients* sont les ordinateurs équipés de navigateurs. Votre ordinateur, équipé d'un navigateur, fait, donc, partie de la catégorie des clients. Chaque client visite un ou plusieurs sites web. Dans les schémas qui vont suivre, le client sera représenté par l'image ci-contre.

Image 1 :

- *Les serveurs* sont des ordinateurs plus ou moins puissants, sur lesquels fonctionnent un serveur web (en 2011, Apache 55 % des serveurs, Microsoft IIS 26 %, Google 8 %) stockent & délivrent des pages de sites web aux navigateurs, c'est-à-dire aux clients.

Une confusion fréquente entre l'ordinateur & le logiciel serveur fait croire qu'un serveur est un ordinateur, alors qu'il s'agit du couple, ordinateur-logiciel serveur. Un même ordinateur puissant peut faire fonctionner plusieurs serveurs web simultanément.

La plupart des internautes n'ont jamais vu un serveur de leur vie. Pourtant, les serveurs sont indispensables au bon fonctionnement du web. L'image ci-contre représentera un serveur dans les schémas suivants.

Image 2 :

La plupart du temps, le serveur est dépourvu d'écran : il reste allumé & travaille tout seul sans intervention humaine, 24h/24, 7j/7.

On résume : votre ordinateur est appelé le client,alors que le client est votre navigateur préféré (Mozilla Firefox, Chromium, Safari, etc.), tandis que l'ordinateur qui détient le site web est appelé le serveur, alors que le serveur est un logiciel, comme (Apache, nginx ou lighttpd, etc.). Comment les deux communiquent-ils ?

C'est justement là que se fait la différence entre un site statique & un site dynamique. Voyons ensemble ce qui change.

CAS D'UN SITE STATIQUE

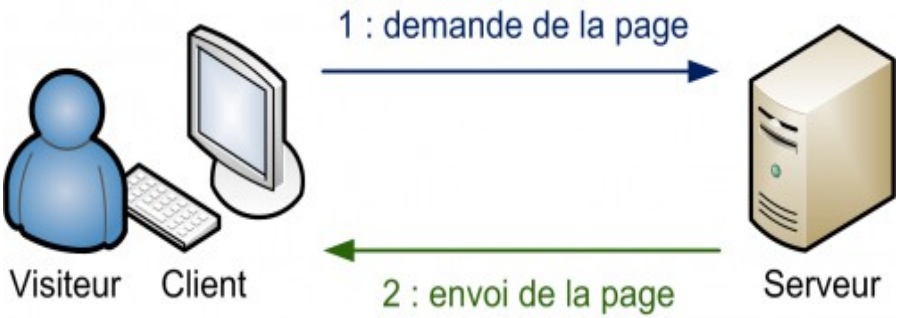

Image 3

Lorsque le site est statique, le schéma est très simple. Cela se passe en deux temps :

1. Le client demande au serveur à voir une page web.
2. Le serveur lui répond en lui envoyant la page réclamée.

La communication est donc simple :

- *Bonjour, je suis le client, pourrais-je voir cette page web ?*
- *Tiens, voilà la page que tu m'as demandée !*

Avec une petite variante, de la part du serveur,

- *Patiente un instant, celle-là, je ne la connais pas il faut que je la demande à un autre serveur !*

Comme le temps d'attente est bref, sauf embouteillage du réseau, cela reste transparent pour le client.

Un site statique se compose de pages web stockées sur le serveur qui les envoie aux clients qui les demandent & qui ne peuvent les modifier.

CAS D'UN SITE DYNAMIQUE

Lorsque le site est dynamique, il y a une étape intermédiaire : la page est générée.

1. Le client demande au serveur une page web ;
2. le serveur qui héberge le site, va générer la page, pour le

client, en allant chercher les informations indispensables dans une base de données relationnelles au moyen d'instructions en langage PHP & finalement en SQL ;

3. puis, il l'envoie au client.

Image 4

La page web est générée à chaque fois qu'un client la réclame. C'est précisément ce qui rend les sites dynamiques vivants : le contenu d'une même page peut changer d'un instant à l'autre mais les pages sont plus longues à afficher.

Étant donné que le serveur génère une page, chaque fois qu'on lui en demande une, il peut la personnaliser en fonction des goûts & des préférences du visiteur [01002], en affichant, par exemple, son pseudonyme sur toutes les pages.

Les Langages de pages statiques

Ce sont les outils servants à décrire les pages webs. Ils sont trois dont deux sont alternatifs : on écrit soit en HTML soit en XHTML. Le troisième, le plus simple est le CSS.

Le CSS (Cascade Style Sheet)

Ce langage est très simple à écrire, mais il permet d'introduire des notions fondamentales pour la compréhension des SBDG & des langages de programmation en général. C'est pour quoi, il sera traité dans l'annexe 4.

LE HTML

Comme tous les produits informatiques, les langages évoluent au cours du temps. Ces évolutions sont repérées par des numéros de version. Les langages ne sont pas, à proprement parler des logiciels : ce sont les interpréteurs & les compilateurs, assurant leur traduction en langage machine qui sont des logiciels. Cependant ils évoluent eux aussi. Afin d'éviter la multiplication de dialectes incompatibles, les acteurs du monde informatique acceptent des normes, qui correspondent aux versions des logiciels. La norme la plus récente du HTML est la norme HTML 5, elle coexiste avec les normes HTML 4.01 Transitional & 4.01 Strict & avec les normes plus anciennes (3, 2 & 1). En d'autres termes, cela veut dire que l'on trouve des sites employant ces différentes normes, peu de créateurs de sites s'amusant à faire évoluer la programmation de leur site, si elle les satisfait, uniquement pour suivre une nouvelle norme.

Voici une page HTML élémentaire :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html lang="fr">
 <head>
  <title>Page d'exemple</title>
 </head>
 <body>
  <p>Bonjour !</p>
 </body>
</html>
```

Une fois saisie dans un fichier terminé par bonjour.html, vous pourrez l'ouvrit dans votre navigateur.

Comme elle n'est pas intégrée à un site, pour l'afficher, il existe une commande Fichier/Ouvrir un fichier... sinon, vous pouvez employer l'URI file:///home/vous/bonjour.html [01003].

Le HTML est un langage avec balises, une balise étant définie comme un mot ou une expression entourée de « < » & de « > ». Sauf exception, les balises vont par paire, une ouvrante & l'autre fermante reconnaissable au « / » qui la débute. Dans cet exemple, les couleurs sont employées pour faire ressortir les paires de balises. Dans le suivant, elles mettent en valeur les éléments constituant le langage : les balises, **les options de balise**, les données textuelles contraintes ou libres, les opérateurs ; c'est la coloration syntaxique.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html lang="fr">
 <head>
  <title>Page d'exemple</title>
 </head>
 <body>
  <p>Bonjour !</p>
 </body>
</html>
```

La première ligne est un commentaire indiquant au traducteur quelle version du langage HTML est employée.

Toute page ouèbe est composée d'un entête (head) & d'un corps (body) ; ces deux parties sont encadrées par la balise html.

Nous allons examiner les balises les plus courantes à travers un exemple réel de page web.

Le site http://www.w3schools.com/TAGS/ contient un descriptif exhaustif des balises & de leurs attributs [01004].

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
```

La première ligne indique la version du HTML employée. Une balise commençant par « <! » est soit un commentaire, elle commence alors par « <!-- » soit un pseudo-commentaire indiquant à

l'interpréteur, la version du HTML. Elle se continue, alors, par une lettre, comme dans le cas présent. Dans le HTML dynamique elle servira à indiquer le langage de script à activer sur le serveur.

```
<html lang="fr">
```

La première balise opérationnelle ; elle n'a que deux attributs **dir**, qui indique le sens de lecture, par défaut de gauche à droite, & **lang**, par défaut "en", raison pour laquelle il faut préciser "fr", pour indiquer l'emploi de notre langue.

```
 <head>
```

Début de l'entête.

```
  <meta http-equiv="Content-Type" content="text/html;
chaF6fet=UTF-8">
```

La balise meta dispose de quatre attributs **name**, **http-equiv**, **scheme** & **content**. Les trois premiers attributs ont des valeurs prédéfinies non modifiables, le dernier est, parfois, laissé à votre discrétion. Certaines valeurs de l'attribut **name** ne sont pas indispensables, comme generator, qui est, généralement, ajouté par le logiciel servant à créer ou à modifier la page concernée. Vous pouvez mettre plusieurs valeurs dans les guillemets, à condition de les séparer par une virgule.

Comme vous pouvez le constater la balise meta ne fonctionne pas en paire. L'annexe I vous indique, pour l'essentiel des balises HTML, l'emploi & le mode d'utilisation de chacune.

```
<meta name="generator" content="Bluefish 1.0.7">
<style type="text/css" media="all">
 @import url(scifo.css);
</style>
```

La balise style peut contenir les définitions des styles de type *css*, mais cela n'a pas grand sens : l'objectif étant d'avoir des styles communs à toutes les pages de façon que la modification d'un style soit répercutée sur toutes les pages, sans avoir à modifier chacune.

En pratique, elle ne devrait contenir qu'une référence à un fichier *css* comme indiqué ci-dessus.

Cette balise n'a qu'un attribut, **media**, qui peut prendre la valeur de :

- all, concerne tous les médias (d'affichage), c'est la valeur par défaut en XHTML ;
- aural, pour les synthétiseurs de parole ;
- braille, pour les appareils tactiles ;
- embossed, pour les imprimantes brailles ;
- handheld, pour tous les dispositifs ayant une limitation quelconque : petitesse de l'écran, monochrome, etc. ;
- print, papier (& tout matériel opaque), & prévisualisations papier sur un écran ;
- projection, vidéo-projecteurs & rétro-projecteurs ;
- screen, écran (d'ordinateurs) couleur, valeur par défaut en HTML4 ;
- tty, médias sur lesquels tous les caractères ont la même taille : télétypes, terminaux, écrans LCD, mobiles... ;
- tv, écrans de télévision (caractérisés par une faible résolution).

```
<title>www.scifo.fr - Avertissement</title>
```

La balise **title** contient le titre qui apparaitra dans la barre de titre de la fenêtre ou dans l'onglet. À ce propos, en HTML les balises peuvent être écrite indifféremment en minuscule ou en majuscule en XHTML, seules les minuscules sont autorisées. Il n'est donc pas inutile de prendre l'habitude de les écrire systématiquement en minuscules.

```
<meta name="verify-v1"
content="D1kJS5bYuULsTg+Ii4UVboYu6PygSXAR4zgv7xnuTto=">
```

La valeur verify-v1 de l'attribut **name** de cette balise **meta** n'est pas standard. Cette ligne a été récupérée sur le site de Google & elle sert à référencer le site.

```
<meta name="keywords" content="r&eacute;flexion, politique,
&eacute;co, socio">
```

La valeur keywords indique aux moteurs de recherche une liste de mot clés permettant de retrouver la page facilement. Les abus d'imbéciles, qui ont employé des mots clés sans rapport avec le contenu de leur page, mais simplement parce qu'ils étaient plus recherché que ceux relatifs au contenu de leur page, font qu'aujourd'hui les moteurs de recherches se fient plus au contenu de la page qu'à ces mots clés, mais certains les emploient encore. La longueur est limitée à mille caractères. À ce propos, vous avez remarque l'écriture réflexion du mot *réflexion*. Si vous n'utilisez pas les équivalents HTML des caractères diacritiques, vous prenez le risque de voir s'afficher, à leur place, des signes cabalistiques dus aux différences d'encodage des caractères. L'annexe II présente la liste de ces équivalents. Attention é compte pour huit caractères, bien qu'il n'en remplace qu'un.

```
<meta name="description" content="Ce site personnel expose
mes r&eacute;flexions &amp; mes multiples activit&eacute;s :
cuisine, jeux de strat&eacute;gie abstraits, informatique,
divertissements.">
```

Cela importe, car la description de la page ne doit pas comporter plus de deux cents caractères.

```
<meta name="author" content="Michel Scifo">
<meta name="copyright" content="Michel Scifo, site sous licence
Creative Commons">
<meta http-equiv="Content-Language" content="fr">
<meta name="robots" content="index,follow">
```

Les attributs de la balise meta sont particulièrement bien décrits sur le site http://www.rankspirit.com/balises.php. Ce site vous aidera à optimiser le vôtre, mais avant de le consulter, il vous faut maîtriser les bases.

```
<script type="text/javascript">
```

```
    var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." :
"http://www.");
    document.write(unescape("%3Cscript F6fc=&rsquo;" + gaJsHost + "google-
analytics.com/ga.js&rsquo; type=&rsquo;text/javascript&rsquo;%3E%3C/script
%3E"));
  </script>
```

Le premier script comme ceux qui suivent il provient du site google-analytics. L'attribut **type** indique le langage du script.

```
  <script type="text/javascript">
   try {
    var pageTracker = _gat._getTracker("UA-10613044-2");
    var pageTracker = _gat._getTracker("UA-11845351-1");
    pageTracker._trackPageview();
   } catch(err) {}
  </script>
 </head>
```

Fin de l'entête.

```
 <body text="#000000" style="background: #ffffff; text-
align:center; height:1000px;">
```

Début de la description de la page. Les attributs indiquent que le texte sera écrit en noir sur fond blanc, qu'il sera centré & que la page aura une hauteur de 1 000 pixels.

```
  <!-- afin de rendre ces lignes compatibles w3c suppression des
blancs autour du "=" (1ère ligne) &amp;amp; remplacement de
l#rsquo ;esperluette par "&amp;" (3e ligne-->
```

Ceci est un commentaire. Quand nous avons testé la compatibilité W3C de cette page ces trois lignes sont apparues en erreur. Il fallut les corrections indiquées pour que le test réussisse. La mise en exposant des abréviations n'est pas possible en HTML, elle a été réalisée dans le traitement de texte.

```
  <script language="javascript" type="text/javascript"> var
stattrak_id=31082; </script>
```

```
<script language="javascript"
type="text/javascript"  rsc="http://stattrak.submitnet.fr/stattrak
.js"></script>
<noscript><img
rsc="http://stattrak.submitnet.fr/cgi/stattrak.pl?
id=31082&amp;nojs=y" height="0" width="0" border="0"
style="display: none" alt=""></noscript>
```

La balise noscript sert, quand un navigateur inhibe, par exemple le javascript, à envoyer un message à l'utilisateur l'avertissant du problème. Ici, un script CGI perl (le langage est indiqué par la terminaison *pl*) est exécuté sur le serveur. L'intéressant est la mise en évidence du passage des paramètres après le caractère ? suivant le nom du script.

```
<div style="background: transparent;
position:relative;width:750px;height:1000px;margin-
left:auto;margin-right:auto;">
```

La balise div sert à grouper des éléments ayant une même fonction, ici la page. Cette balise comporte deux sortes d'attributs : les standards, comme celui figurant dans la balise ci-dessus [01005] & les évènementiels, liés à un évènement (par exemple : clic ou déplacement de la souris, appui ou relâchement d'un de ses boutons ou d'une des touches du clavier, etc.)

```
<!-- haut -->
    <!-- Mappemonde -->
    <div style="position:absolute; left:455px; top:0px;
width:295px; height:139px;">
```

Définition d'une première zone qui va contenir la mappemonde présentant les zones désertiques de notre planète.

```
    <img rsc="images/acc_mon.png" width="295" height="139"
border="0" id="pic_20" name="pic_20" title="" alt="">
```

La balise img permet d'insérer une image en précisant ses dimensions & ses paramètres.

```
</div>
<!-- Le ma&icirc ;tre r&eacute;fleur -->
<h1>
```

La balise h1 est une des six balises de présentation de titres incluent dans ce langage. Elles peuvent être redéfinies dans la feuille de style *css*.

```
<div style="position:absolute; color:#007000;text-
decoration:none;" left:40px; top:30px; width:410px;
height:100px;">
```
01016

Définition d'une seconde zone qui va contenir le nom du site. Cette zone contenant du texte, celui-ci sera de base en style h1.

```
Le <span style="color:#ba0002;"> ma&icirc;tre </span><span
style="color:#007000;">r&eacute;fleur</span>
```

La balise span sert généralement à modifier un groupe de caractères à l'aide d'options précisées dans la balise de début. L'attribut **class** permet d'employer des styles définis dans le fichier *css*. L'attribut **style** contient une chaîne de caractères dans lesquels différents autres attributs sont valorisés en les séparant par un ;.

```
</div>
</h1>
<h1>
<div style="position:absolute; left:40px; top:100px;
width:410px; height:34px;">
Une troisième zone contient le sous-titre.
<span style="font-family:IM FELL French Canon, serif; font-
size:27.0px; font-variant:small-caps;"><span
style="color:#ba0002;">Libre</span><span
style="color:#007000;"> penseur</span><span
style="color:#de0026;"> pansu</span><span
style="color:#00b800;"> terrien</span>
</span>
```

Vous le constatez, il est possible d'imbriquer les balises span.

```
</div>
</h1>
```

Vous devez maintenant être capable de commenter les lignes suivantes.

Une partie du fichier est omise afin d'arriver aux nouveautés.

```
<!-- menu du haut -->
```

Dans cette page, le menu a été réduit à sa plus simple expression, car elle conseille, seulement, aux utilisateurs d'Internet Explorer de changer de navigateur, pour parcourir ce site.

```
<div id="mh" style="position:relative; left:0px; top:173px;
width:750px; height:20px;">
```

Première remarque, c'est la première zone ayant un attribut id.

```
<h3 style="font-family:Brassiere;">
```

Cette fois empli du style h3.

```
<a style="color:#ba0002; text-decoration:none;"
href="index.html">Bienvenue</a>
```

La balise a sert à établir des liens soit avec d'autres pages html, soit avec des repères créés avec cette même balise dans la même page. Elle possède un attribut **name**, inutilisé ici & un attribut **href**. La valeur de **href** est soit un nom de ficher html, soit l'attribut **name** d'une autre balise **a**, si le lien est dans la même page.

Ici, des lignes omises.

```
<div id="txt_73" style="position:absolute; left:36px; top:325px;
width:555px; height:451px;-moz-box-sizing:border-box;box-
sizing:border-box; overflow:auto;">
```

Cette zone-là pouvant accueillir plus de texte qu'elle ne peut en afficher, il est demandé d'afficher des ascenseurs, si la longueur du texte dépasse la longueur de la zone.

```
<p class="alisuiv">Ce site a &eacute;t&eacute; conçu en respectant,
strictement, les pr&eacute;conisations de programmation du W3C,
```

l’organisme définissant les règles d’Internet. Chacune des pages a été validée sur le site de cet organisme, avant de recevoir les séquences de code nécessaires à Google pour établir les statistiques de fréquentation. </p>

Premier paragraphe du texte. La balise p délimite les paragraphes, son attribut **class**, le style du paragraphe.

Ici, des lignes omises.

```
 </body>
</html>
```

Fin de la page.

Avec ces seules balises & leurs attributs, vous disposez de suffisamment d'éléments pour réaliser un site statique de qualité.

Le XHTML

Le XHTML est une extension du HTML, le rendant plus commode à employer, particulièrement au niveau du traitement des données. En contrepartie, il nécessite un formalisme plus complexe.

La page suivante donne le même résultat que la page bonjour.html. Vous pourrez la télécharger à l'adresse : ftp://www.scifo.fr/info/web/bonjx.html.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xml:lang="fr">
 <head>
  <title>Page d&apos;exemple</title>
  <meta  http-equiv="Content-Type"  content="text/html; charset=utf-
8"  />
 </head>
 <body>
```

```
  <p>Bonjour !</p>
 </body>
</html>
```

L'annexe 3 présente les différences entre HTML & XHTML.

LE PHP & LES PAGES DYNAMIQUES

Voici une partie de la première page d'un site web dynamique créé avec Wordpress.

LE HTML GÉNÉRÉ

```
<!DOCTYPE html>
<html lang="en-US" class="no-js">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width">
 <link rel="profile" href="http://gmpg.org/xfn/11">
 <link rel="pingback" href="http://127.0.0.1/wordpress/xmlrpc.php">
 <!--[if lt IE 9]>
 <script src="http://127.0.0.1/wordpress/wp-content/themes/twenty-fifteen/js/html5.js"></script>
 <![endif]-->
```

```
    <script>(function(){document.documentElement.className='js'})
();</script>
    <title>michel&#039;s Blog! | Just another WordPress site</title>
   <link rel="alternate" type="application/rss+xml" title="michel&#039;s
Blog! &raquo;"[...]/>
   <link rel="alternate" type="application/rss+xml" title="michel&#039;s
Blog! &raquo; [...]/>
   <link rel='stylesheet' id='open-sans-css'
href='//fonts.googleapis.com/css?family=[...] />
   <link rel='stylesheet' id='dashicons-css'  href='http://127.0.0.1/word-
press/[...]
   <link rel='stylesheet' id='admin-bar-css'  href='http://127.0.0.1/word-
press/[...]
   [... 22 lignes]
   </style>
   </head>

   <body class="home blog logged-in admin-bar no-customize-support">
   <div id="page" class="hfeed site">
    <a class="skip-link screen-reader-text" href="#content">Skip to
content</a>
     <div id="sidebar" class="sidebar">
      <header id="masthead" class="site-header" role="banner">
       <div class="site-branding">
                  <h1 class="site-title"><a href="http://127.0.0.1/wordpress/"
rel="home">michel&#039;s Blog!</a></h1>
                  <p class="site-description">Just another WordPress
site</p>
                <button class="secondary-toggle">Menu and widgets</but-
ton>
   [... 130 lignes]
```

```
</body>
</html>
```

LA LISTE D'INDEX.PHP

```
<?php
/**
 * Front to the WordPress application. This file doesn't do anything, but loads wp-blog-header.php which does and tells WordPress to load the theme. [...]
 */
define('WP_USE_THEMES', true);

/** Loads the WordPress Environment and Template */
require( dirname( __FILE__ ) . '/wp-blog-header.php' );
```

Notez que le script ne se termine pas par le symbolee ?>, il figurera dans le dernier script appelé.

LES DONNÉES SAISIES

Il s'agit d'un très court extrait d'une des lignes d'une des onze tables de la base de données de Wordpress.

Résultat de la requête SQL

Client : localhost
Base de données : bitnami_wordpress
Généré le : Mar 13 Janvier 2015 à 13:06
Généré par : phpMyAdmin 4.2.11 / MySQL 5.6.21
Requête SQL : SELECT * FROM `wp_posts` LIMIT 0, 25 ;
Lignes : 4

post_date	post_content	post_title

2015-01-13 11:34:29	Site réalisé avec Wordpress sans programmer une ligne !	Bonjour !

Notions de SGBD

Remarque : si vous êtes plus pragmatique que conceptuel, la lecture de ce chapitre section peut se faire après celle du chapitre MySQL/MariaDB, mais elle nous semble indispensable pour comprendre le contexte de travail.

Un **s**ystème de **g**estion de **b**ase de **d**onnées (SGBD) est un ensemble de programmes informatiques qui contrôle la création, l'entretien & l'utilisation de bases de données. Il permet aux organisations de placer le contrôle du développement de bases de données dans les mains des administrateurs de base de données (DBA) & d'autres spécialistes. C'est un logiciel qui permet l'utilisation de la collection intégrée des enregistrements de données & des fichiers connue sous le nom de bases de données.

Un SGBD comporte différents outils pour effectuer des opérations sur les bases de données. Ces outils sont plus ou moins sophistiqués suivant l'usage & le public cible du SGBD : composant logiciel, serveur, logiciel applicatif, développeur, administrateur de bases de données, tout public.

Le problème majeur des SGBD s'avère leur incompatibilité. Pour remédier à ce problème Microsoft à une idée originale, la seule, à ce jour de 2011, c'est toujours vrai en 2014, qu'ils aient eu : créer un logiciel permettant de gérer des bases de données indépendamment du SGBD ayant servi à les concevoir : ODBC, par l'intermédiaire d'une interface de programmation unique, depuis imitée par d'autre comme JDBC, etc. Ce sont des logiciels souvent utilisés avec les SGBD.

Revenons aux SGBD, non seulement ils sont incompatibles entre eux mais en plus ils ont des modes de fonctionnement différents.

* Les composants logiciels tel que *SQLite* sont utilisés par des logiciels tiers, l'interface de programmation est sophistiquée, les interfaces utilisateurs sont minimales voire inexistantes. Ils sont dits SGBD embarqués s'ils sont incorporés dans des logiciels tiers.
* Dans les *clients-serveurs* de base de données, tels *Oracle Database*, ou *MariaDB*, les serveurs répondent à des demandes provenant de clients d'un réseau informatique. Les clients sont des logiciels tiers. L'interface utilisateur sert essentiellement à imposer des règles tels que des listes de contrôle d'accès en vue de limiter ou d'interdire certaines opérations & ainsi assurer la sécurité & la confidentialité du contenu des bases de données. Le SGBD est équipé d'une interface de programmation sophistiquée prévue pour être utilisée en même temps par différents ordinateurs d'un réseau informatique.
* Les applicatifs comme *Ms-Access* permettent à des utilisateurs d'enregistrer, d'organiser, & de retrouver un ensemble d'informations sans recours à des logiciels tiers & sans connaissance technique. Une interface-utilisateur sophistiquée permet à un utilisateur d'effectuer de nombreuses opérations de manipulation de données, cette interface est souvent en style *Query By Example*. L'interface de programmation est minimale voire inexistante.
* Les serveurs, en voie de disparition, tels *dBase* ou *Paradox* étaient des environnements de développement intégrés permettant de créer de petits logiciels applicatifs de base de données avec peu de connaissances techniques & un code source plus ou moins masqué. Ils fonctionnaient en mono-poste, puis se sont adaptés aux petits réseaux, en évoluant vers un fonctionnement client-serveur peu optimisé. Seul *MS-Access* soutenu par le marketing de Microsoft & par une excellente ergonomie, bien que nettement moins performants que les deux précités a pu évoluer vers le type *Applicatif*.

Selon le cabinet Gartner Dataquest, le marché des SGBD représente en 2004 plus de 15 milliards de dollars, en augmentation de 7 %. 80 %

des parts sont pour les SGBD relationnels. Les trois ténors du marché, IBM DB2, Oracle & Microsoft SQL Server occupaient 85 % du marché.

Les SGBD gratuits ne sont pas en reste. Selon Sleepycat Software : il existe plus de 200 millions de copies de Berkeley DB (SGBD standard des unix) en circulation ; MySQL/MariaDB a été téléchargé (en 2011) plus de 6 millions de fois ; & PostgreSQL plus de 1 million de fois.

Voici un tableau présentant quelques SGBD connus.

SGBD	Année	Éditeur	Utilisation	Licence
Pick	1968	pick system	serveur	propriétaire
MaxDB	1977	SAP AG	composant	GPL
INGRES	1977	INGRES	CS	propriétaire
dBase	1978	Ashton-Tate	serveur	propriétaire
Oracle	1979	Oracle Corporation	CS	propriétaire
DB2	1983	IBM	CS	propriétaire
4eDimension	1985	4eDimension	applicatif	propriétaire
PostgreSQL	1985	Michæl Stonebraker	CS	BSD
Berkeley DB	1986	Université de Berkeley	CL	BSD
Paradox	1987	Corel	serveur	propriétaire
SQL Server	1989	Microsoft	CS	propriétaire
Access	1992	Microsoft	applicatif	propriétaire
MySQL	1995	Oracle Corporation	CS	GPL
Derby	1996	Apache Software Foundation	composant	Apache
SQLite	2000	D. Richard Hipp	composant	DP
OpenOffice.org Base	2002	Oracle Corporation	applicatif	LGPL

SGBD	ANNÉE	ÉDITEUR	UTILISATION	LICENCE
MariaDB	2009	Monty Program Ab	CS	GPL

Le nom de l'éditeur est celui du dernier éditeur connu. *MariaDB* est une copie conforme de *MySQL* créée après le rachat de Sun MicroSystems par Oracle Corporation, éditeur du SGBD éponyme.

ORGANISATION D'UN SYSTÈME DE GESTION DE BASE DE DONNÉES

Dans les grands systèmes les SGBD permettent aux utilisateurs (humains ou logiciels) de stocker & de retrouver leurs données de façon structurée, en écrivant leurs demandes dans un langage d'interrogation du SGBD. De plus, les SGBD modernes permettent de construire de véritables applications [0200].

Ils permettent à différentes applications d'accéder facilement à la même base de données. Pour ce faire ils utilisent un modèle de base de données.

Il y a six grands modèles.

- Le *modèle plat* se compose d'un seul tableau de données à deux dimensions ; c'est le cas des bases de données dans les tableurs ; toutes les cellules d'une colonne donnée contiennent des valeurs supposées similaires, & toutes les cellules d'une ligne sont supposés être liées les unes aux autres (données au format csv par exemple) ; il est peut-être abusif, dans ce cas, de parler de base de donnée.
- Le *modèle hiérarchique* organise les données en une structure de type arbre, ; en clair chaque enfant n'a qu'un parent, tout comme sur un arbre chaque feuille ne peut se trouver que sur une seule branche ; ce modèle n'est plus utilisé pour gérer les bases de données ; en revanche il est crucial pour les systèmes de gestion de ficher & il sert à structurer le XML ; cette structure est très efficace

pour décrire de nombreuses relations dans le monde réel : recettes, table des matières, etc.

* Le *modèle réseau* est une évolution du modèle hiérarchiques, permettant d'une part de gérer plusieurs parents pour un enfant & d'autre part de remonter dans la hiérarchie ; dans les années 1970, il offrait des performances équivalentes ou supérieures à celles du modèle relationnel, mais la progression des performance matérielles, la croissance exponentielle des données ont favorisé le modèle relationnel plus flexible.
* Le *modèle relationnel*[02002] vise d'une part, d'organiser les données en tables liées entre elles par des données communes afin de : fournir un meilleur contrôle de l'accès aux données, de mieux assurer leur intégrité (ne pas supprimer une fiche ***parents*** si des fiches ***enfants*** existent, par exemple), faciliter les accès concurrents (pseudosimultanés) aux données & leurs sauvegardes, & d'autre part, de fournir un langage d'interrogation du SGBD évitant d'écrire des programmes pour en extraire des informations.
* Le *modèle relationnel objet* est un modèle relationnel de base de données, mais gérant des objets, des classes & l'héritage ; sa complexité de mise en œuvre le limite à des domaines eux-mêmes très complexes, par exemple, la biologie moléculaire.
* Le *modèle entrepôt de données*[02003] permet d'organiser les données dans les dits entrepôts, en combinant un schéma en étoile (une table en référence plusieurs autres, cas particulier du schéma en flocon) & une évolution du modèle relationnel appelée modèle dimensionnel.

La réalité étant, parfois, complexe, il arrive qu'il faille combiner les modèles pour l'analyser ; ainsi Internet fonctionne globalement, selon un modèle réseau & localement, selon un modèle hiérarchique.

Nous nous intéresserons aux SGBD relationnels & plus particulièrement à *MySQL/MariaDB*.

Ce SGBD fonctionne donc avec des clients travaillant avec un serveur. Le mot serveur désigne ici un logiciel, mais ce mot désigne aussi l'ordinateur sur lequel fonctionne ce logiciel.

Si un simple PC peut servir une base de données de plusieurs gigaoctets accessibles simultanément par une dizaine d'utilisateurs, la plus grosse base de données du monde, celle de Google[02004], nécessite près de deux millions de PC biprocesseurs avec, chacun, 2 ou 4 Go de mémoire & 80 Go sur des disques durs redondants, interconnectés à très hauts-débits, regroupés par 1000, dans des caissons réfrigérés, installés en douze emplacements dans le monde. Plus modestement dans les PME, on emploie des serveurs multiprocesseurs équipés de plusieurs gigaoctets de mémoire & de systèmes de stockage sécurisés afin d'éviter une interruption d'activité ou des pertes de données suite à des pannes.

Même dans une très petite structure, il faut assurer la continuité du service & éviter les pertes de données qu'elles soient dues à des pannes ou à de la malveillance.

Les premiers SGBD relationnels System-R d'IBM & INGRES de l'université de Berkeley fonctionnaient sur des serveurs départementaux, relativement isolés, & peu nombreux. Aujourd'hui, des millions de serveurs étant actifs, les problèmes de sécurités sont plus graves.

Le concept de base de données relationnelles a été inventé par Edgar Codd un ingénieur d'IBM en 1970. Son idée a été reprise & développées par deux chercheurs de l'université de Berkeley, Eugene Wong & Michael Stonebraker. Chez IBM naquit R-system dont le descendant se nomme Oracle. À Berkeley naquit INGRES dont descendent MySQL/MariaDB & PostgreSQL (réécriture du SGBD Sybase, inspiré directement d'INGRES), entre autres.

C'est IBM qui a imposé le SQL bien qu'il viole plusieurs principes de fonctionnement du modèle relationnel.

Depuis deux ans sont apparus des SGBD dits *No-SQL* (*Not Only SQL*), un peu moins rapide, mais plus conformes au modèle.

Fonctionnement d'un SGBD dit relationnel

Un SGBD relationnel actuel est composé de quatre parties, un langage de modélisation des données, une structure de données, un langage d'interrogation & un mécanisme de transaction [02005]. Ces parties intègrent :

- un ou plusieurs *moteurs de base de données*, qui définissent l'organisation du stockage sur les mémoires de masse & l'accès à ces mémoires afin de répondre aux requêtes ;
- un *sous-système de définition des données* permettant au concepteur de créer & de maintenir le dictionnaire des données, ce qui évite de redéfinir une données & facilite la création des tables &des index ;
- un *sous-système de manipulation des données* permettant l'ajout, la modification la consultation & la suppression des données dans les tables ;
- un *sous-système de génération d'application* comprenant un utilitaire de définition des tables, un de création des formulaires de saisie & un générateur de rapport ;
- un *sous-système d'administration des données* optimisant les requêtes, assurant, parfois, la sauvegarde & la restauration.

Comme tous les SBGD performants, à l'exception de celui d'IBM ont été conçus sur des serveurs Unix, ils intègrent une gestion des utilisateurs bien plus sophistiquée que celle de ce système d'exploitation, pouvant traiter près d'une quarantaine de droits divers (droits de : créer une base de données, de la lire, de l'effacer, idem pour les tables, les lignes, les colonnes, etc.).

Le langage de modélisation des données

Il est bien entendu de type relationnel. Certains SGBD autorise l'emploi d'autres langages de modélisation de façon annexe. Cependant, le modèle dominant aujourd'hui basé sur le SQL n'est pas purement relationnel.

ODBC est souvent compatible avec les SGBD, c'est une interface permettant d'accéder à d'autres langages de modélisation des données qu'ils soient proches (autre SGBD SQL) ou différents (modèle plat pour le format csv par exemple).

STRUCTURE DE DONNÉES

Les structures de données sont les colonnes, les lignes, les tables & autres objets intégrés dans une base de données. Ces structures sont optimisées pour gérer des tables de grande taille.

LANGAGE D'INTERROGATION DE BASES DE DONNÉES

Joint à un générateur d'état, ils permettent d'interroger interactivement une BD, d'analyser les résultats de cette interrogation & en particulier de déceler des erreurs de saisie à corriger. Les privilèges utilisateurs autorisent la mise en place de mesures de sécurités en restreignant les droits de consultation & de modification de façon adaptée aux besoins.

LA GESTION DES TRANSACTIONS

Un mécanisme de gestion des transactions doit être *ACID* [02006]. En clair il doit posséder les quatre propriétés suivantes :

Une transaction doit respecter les quatre contraintes suivantes dites *ACID* :

- *atomicité* (A), une transaction doit s'effectuer en tout ou rien, c'est-à-dire complètement ou pas du tout ;

- *cohérence* (C), La cohérence des données doit être assurée dans tous les cas, même dans les cas d'erreur où le système doit revenir au précédent état cohérent ;
- *isolation* (I), la transaction va travailler dans un mode isolé où elle seule peut voir les données qu'elle est en train de modifier, cela en attente d'un nouveau point de synchronisation ; le système garantit aux autres transactions, exécutées en parallèle sur le même système, une visibilité sur les données antérieures ;
- *durabilité* (D), Lorsque la transaction est achevée, le système est dans un état stable durable, soit à l'issue d'une modification transactionnelle réussie, soit à l'issue d'un échec qui se solde par le retour à l'état stable antérieur.

Exemples

- Une opération immobilière de vente d'appartement peut durer longtemps. Informatiquement, elle sera considérée comme un tout composé de plusieurs transactions informatiques : réservation, proposition d'achat, compromis, acte notarié (on appelle ce tout une procédure fonctionnelle). Les étapes intermédiaires sont donc gérées via l'état de l'objet Appartement. Même si la procédure est en cours, entre chaque étape le système reste cohérent. En cas d'abandon de la procédure, les règles fonctionnelles remettront l'appartement en vente.
- La saisie comptable d'une facture est également une transaction : elle concerne plusieurs comptes, génère plusieurs écritures : vente, client, TVA perçue, banque ou caisse, le grand livre, le livre de banque ; si l'une des écritures ne peut s'effectuer la transaction sera annulée & un message d'erreur envoyé à l'opérateur.

L'objectif des propriétés *ACID* est de maintenir l'intégrité des données, but auquel le SGBD travaille, également, avec d'autres outils, verrouillage des fiches en cours de saisie, prévention de la redondance des informations, etc.

Nous pouvons maintenant concrétiser ces notions en examinant de le fonctionnement d'un SGBD précis *MySQL/MariaDB*.

Image 1


XAMPP
5.6.3
[PHP: 5.6.3]
Bienvenue
Statut
Sécurité
Documentation
Composants
Applications
Php
phpinfo()
Collection de CD
Biorhythme
Instant Art
Répertoire Téléphonique
Perl
perlinfo()
Guest Book
J2ee
Info
Tomcat examples
Tools
phpMyAdmin
FileZilla FTP
Webalizer
Mail
©2002-2015
...APACHE FRIENDS...

Image 2

MySQL/MariaDB

L'utilisation d'un SGBD Relationnel contemporain se fait au moyen de requêtes SQL. Après avoir présenté les différents logiciels qui composent ce SGBD, le client, le serveur, les utilitaires d'administration & l'utilitaire ODBC, nous présenteront le SQL, puis les différents moteurs de bases de données.

Auparavant, il nous faut installer un serveur web, le serveur *MariaDB* ou *MySQL* & l'interpréteur *PHP*.

Pour commencer, nous allons nous simplifier le travail en installant *xampp*, qui intègre ces trois composantes, entre autres.

Nous verrons, ensuite, comment installer un serveur *Apache* & le serveur *MariaDB*.

XAMPP

Le site officiel du logiciel se nomme www.apachefriends.org. Une fois téléchargée, la version Windows (win32-5.6.3-0-VC11-installer.exe) ou la version Linux (xampp-linux-x64-5.6.3-0-installer.run) exécutez cet installateur en validant les choix par défaut ! En cliquant sur Finish vous verrez apparaître cette fenêtre (Linux) :

Image 1

ou celle-ci (Windows).

Nous allons nous intéresser à cette dernière, plus complexe [03001]. Les boutons sur la droite concernent le paramétrage global de *xampp*. Les boutons sur les lignes commençant par un nom de serveur ne regardent que le dit logiciel. Ni PHP ni Perl n'y apparaisse, puisque ce ne sont pas des serveurs.

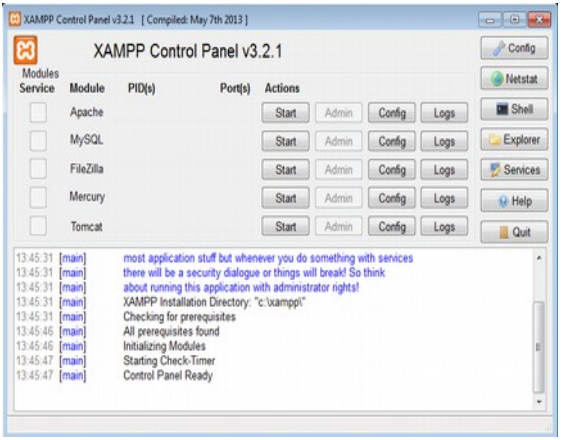

Image 2

Nous allons démarrer *Apache* & *MySQL*. Si par hasard vous avez déjà un serveur *Apache* ou *MySQl* actifs sur votre PC, leur bouton [Start] s'intitule Stop.

Activez ces deux serveurs en cliquant sur le bouton [Start] correspondant, puis répondez aux sollicitations du *pare-feu windows*, s'il est actif.

Pour employer *xampp*, il vous faut maintenant, lancer votre navigateur préféré & saisir localhost ou 127.0.0.1 dans la barre d'adresse.

Si vous ne voyez pas apparaître ce qui suit au milieu de la fenêtre du navigateur, vous avez un problème.

Choisissez la langue. Ce sera Français pour nous.

Le menu de *xampp* comporte cinq parties, dans l'ordre inverse, il s'agit de :

- *Tools* propose quatre outils, trois que nus n'emploierons pas dans le cadre de ce cours & phpMyAdmin que nous solliciterons intensivement ;
- *J2ee* permet de travailler en Java, ce que nous ne ferons pas ;

- *Perl*, fait de même pour ce langage & nous ne l'emploierons pas plus ;
- *PHP* en revanche sera plus sollicitée ; la première ligne apporte des informations sur l'interpréteur, les autres sont des applications exemples ;
- la *première partie*, propose :
  * dans *Applications*, un lien vers le site bitnami.com. C'est un autre des sites de la société BitRock qui édite xampp & le site appachafriends.com ; ce site vous propose des versions d'une cinquantaine de logicels libres & performants ;
  * dans *Composants*, la liste des modules de xampp ;
  * *Bienvenue* & *Documentation* se passent de commentaires ;
  * *Statut* vous indique quels modules sont activés ou désactivés ;
  * Enfin, *Sécurité* [03002] vous informe que ni *xampp*, ni *MySQL*, ni *phpMyAdmin* ne sont sécurisés, ou en d'autres termes qu'aucun mot de passe n'a été défini pour ces logiciels. Si un serveur ftp est actif, il l'inclura dans la liste. Seul le début de cette page est en français, mais la seule chose à comprendre, pour l'instant, c'est que vous devez cliquer sur l'hyperlien indiqué afin de saisir les mots de passe.

Une fois les mots de passe saisis, si vous recliquez sur Sécurité, tous les voyants seront verts !

phpMyAdmin

Vous pouvez maintenant activer phpMyAdmin, le client MySQL, & vous connecter en tant qu'utilisateur administrateur MySQL (root). À partir de maintenant, il n'y a pratiquement plus de différences entre les versions Windows & Linux de *xampp*.

L'écran de démarrage de ce client comporte cinq parties.

* *La première*, comporte en dessous du nom du client, un menu iconique, puis la listes des bases de données. Lors de la première activation de *phpMyAdmin*, il n'y a aucune base récemment ouverte ou préférée !

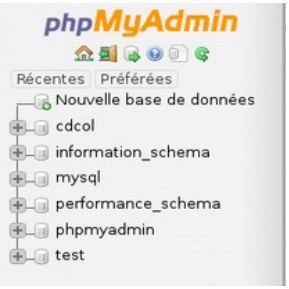

Image 6

Cdcol est une base de donnée d'exemple. Avec *MySQL* une base de données est un dossier contenant des fichiers appelés tables (les données) ou index (des informations pour accélérer la recherche des données). Nous y reviendrons.

* *La seconde* partie présente le menu, quasi-complet puisque la fin du menu se trouve sous le bouton plus.

Image 7

Pour l'instant nous ne nous intéresserons pas aux commandes SQL, Utilisateurs, Exporter, Paramètres, Réplication, Variables & plus.

Même si la commande Importer nous importe, nous ne la traiterons qu'avec l'abord de notre base exemple. En effet, plutôt que de saisir toutes les définitions des tables de la base, nous saisirons la première & nous importerons les autres, ainsi que leurs éventuelles données.

La commande État, fondamentale pour la maintenance, affiche un sous-menu suivi d'informations sur les transferts de données entre le serveur & le client.

Image 8

* *La troisième* partie fournie affiche quelques-uns des paramètres généraux du client, comme l'interclassement choisi par défaut, nous y reviendrons.

Par défaut, c'est le thème pmahomme qui est sélectionné. Comme nous le trouvons moins agréable que le thème Original nous l'avons changé.

Comme son nom l'indique Plus de paramètres affiche un écran vous permettant de fournir des paramètres supplémentaires, mais comme cet écran risque de n'avoir aucun sens pour vous, évitez, pour l'instant, de l'activer.

* La quatrième tout en bas de l'écran, peut être ignorée pour le moment. Ne cliquez pas sur le lien ici, sous peine de voir apparaître une page dont le contenu vous serait totalement incompréhensible.
* La dernière, sur la droite de l'écran affiche des informations sur le serveur *MySQL*, le serveur *Apache 2* & *phpMyAdmin*.

La première ligne, vous indique que cette copie d'écran a été faite sur le PC servant à rédiger ce document & que ce PC utilise un Unix. La deuxième & la troisième précisent le nom du SGBD. La cinquième vous indique quel utilisateur vous êtes & avec quel ordinateur vous vous connectez, ce qui ne présente aucun intérêt. En revanche, l'information de la sixième ligne est fondamentale si vous utilisez, comme il se doit, les caractères diacritiques (caractères avec accents, cédille, tilde, macron, etc.). Les informations sur le serveur web ne devrait pas vous parler beaucoup pour le moment, mais les deux premiers liens relatifs à *phpMyAdmin* pourront vous aider par la suite.

Remarque ; ce que nous allons décrire maintenant est indépendant du système, Windows *ou* Linux*, avec ou sans xampp, avec ou sans phpMyAdmin, rien ne changera.*

La première chose à faire pour pouvoir utiliser *MariaDB* est de créer la base de donnée que nous souhaitons employer. Avec la plu-

part des logiciels (*phpMyAdmin*, *Wordpress*, *Magento*, *Drupal*, etc.) cela se fait automatiquement durant la phase d'installation. La seule chose que l'on vous demande, parfois, est le nom de la base.

Le SGBD, tout comme le client graphique nécessite des base de données, *MySQL* en emploie trois bases : *mysql*, *information_schema* & *performance_schema*. La base test n'est pas indispensable, elle sert d'exemple, la base *cd_col* a été rajoutée par *xampp* & la dernière, vous l'avez deviné par *phpMyAdmin*. Nous allons en rajouter une que nous baptiserons *dvd*. L'auteur de MySQL étant suédois, cela explique le choix de cette langue pour le classement des caractères.

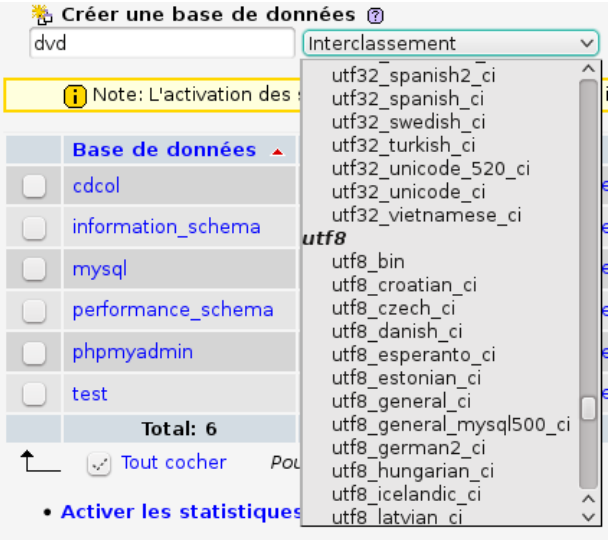

Image 9

L'interclassement est l'ordre dans lequel on range les caractères. Par défaut l'ordre est celui du code ASCII dans lequel Z est classé avant a & z avant à. L'interclassement range toutes les lettres ayant un signe diacritique au même niveau que la lettre qui n'en porte pas (exemple : A, a, À, à ; â, Â). L'interclassement utf8_international_ci convient parfaitement à toutes les langues européennes & aussi au turc.

La création d'une base & de ses tables est du ressort du concepteur de base de données plus que de l'administrateur. Cependant savoir en quoi consiste ce travail peut aider à l'administrer. De plus, nous allons pouvoir nous livrer à un acte d'administration en cli-

quant sur Vérifier les privilèges. Nous y reviendrons dans les travaux pratiques.

Il est temps de s'intéresser aux composants de ce logiciel de gestion de bases de données.

Exxercices

Ex. 01

Ex. 02

Ex. 03

Ex. 04

Les Composants de MySQL/MariaDB

Nous ne présenterons pas, ici, tous les composants, simplement les principaux. Les autres le sont dans l'annexe x.

Le Serveur

Étant issu du monde Unix, il s'appelle **mysqld**. Le « d » final correspondant au mot *démon*, nom dont ont affuble les *services* dans les unix.

Comme tous les logiciels complexes, il doit être configuré de façon à fonctionner de façon optimale. Cette configuration peut se faire soit au moyen d'un fichier de configuration (my.ini,avec Windows, my.cnf avec Linux) soit en ligne de commande. Le fichier complet & la liste des options possibles pour le serveur se trouvent en annexe S05.

Les lignes en rouges sont à modifier, celles en vert indique des valeurs chaudement recommandées, nos commentaires sont dans la police usuelle.

Voici le début du fichier my.ini créé lors de l'installation avec Windows. Les lignes vides ont été supprimées. Les commentaires ont été regroupés par paragraphes.

```
# MySQL Server Instance Configuration File
# ----------------------------------------------------------------------
# Generated by the MySQL Server Instance Configuration Wizard
Installation Instructions
# ----------------------------------------------------------------------
# On Linux you can copy this file to /etc/my.cnf to set global options,
mysql-data-dir/my.cnf to set server-specific options (@localstatedir@
for this installation) or to ~/.my.cnf to set user-specific options.
# On Windows you should keep this file in the installation
directory of your server (e.g. C:\Program Files\MySQL\MySQL Server
X.Y). To make sure the server reads the config file use the startup
option "--defaults-file".
[...]
```

En pratique, si lors de l'installation vous avez coché *installer comme un service*, ceci est inutile, particulièrement si n'utilisez *MySQL/MariaDB* qu'après avoir redémarré votre PC.

```
# Guildlines for editing this file
[...]
# CLIENT SECTION
# ----------------------------------------------------------------------
[...]
[client]
port=3306
[mysql]
default-character-set=utf8
```

Chaudement recommandé, cela ne ralentit ni le serveur ni le client & cela facilite les transferts de données entre systèmes d'exploitation & le traitement des langues.

```
# SERVER SECTION
# ----------------------------------------------------------------------
[...]
#
[mysqld]
# The TCP/IP Port the MySQL/MariaDB Server will listen on
port=3306
#Path to installation directory. All paths are usually resolved relative
to this.
basedir="C:/Program Files (x86)/MySQL/MySQL Server 5.1/"
#Path to the database root
datadir="C:/ProgramData/MySQL/MySQL Server 5.1/Data/"
```

Il vaut mieux indiquer ici le dossier dans lequel vous souhaitez installer les bases, de préférence sur un autre disque logique ou mieux sur un autre disque physique.

```
# The default character set that will be used when a new schema or
table is created and no character set is defined
default-character-set=utf8
# The default storage engine that will be used when create new
tables when default.
storage-engine=INNODB
# Set the SQL mode to strict
sql-mode="STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,\
NO_ENGINE_SUBSTITUTION"
[...]
```

Le caractère \, en fin de ligne, indique que la ligne se continue à la ligne suivante !

La liste des options disponibles pour le serveur (19 pages A4) s'obtient en tapant mysqld --verbose --help dans une console.

```
mysqld Ver 5.1.49-community for Win32 on ia32 (MySQL Community
Server (GPL))
Copyright (C) 2000-2008 MySQL AB, by Monty and others.
Copyright (C) 2008 Sun Microsystems, Inc.
[...]
Starts the MySQL database server.
Usage: mysqld [OPTIONS]
NT and Win32 specific options:
 --install              Install the default service (NT).
 --install-manual       Install the default service started manually (NT).
 [...]
 --remove               Remove the default service from the service list
(NT).
  [...]
```

ou avec unix

```
mysqld  Ver 10.0.13-MariaDB for Linux on x86_64 (openSUSE package)
Copyright (c) 2000, 2014, Oracle, SkySQL Ab and others.

Starts the MariaDB database server.

Usage: mysqld [OPTIONS]

Default options are read from the following files in the given order:
/etc/my.cnf ~/.my.cnf
The following groups are read: mysqld server mysqld-10.0 mariadb
mariadb-10.0 client-server
The following options may be given as the first argument:
--print-defaults        Print the program argument list and exit.
[...]
```

Rappel, l'installation sous forme de service est automatique, avec l'option démarrage automatique. Si vous employez *MySQL/MariaDB* tous les jours, il ne vous faut rien changer. En revanche, si votre utilisation n'est qu'occasionnelle, il vous faudra copier dans un fichier bat, l'ensemble des options avec lesquelles vous souhaitez que le serveur démarre.

```
Default options are read from the following files in the given order:
C:\Windows\my.ini C:\Windows\my.cnf C:\my.ini C:\my.cnf C:\Program
Files (x86)\MySQL\MySQL Server 5.1\my.ini C:\Program Files
(x86)\MySQL\MySQL Server 5.1\my.cnf
```

En principe my.ini est le nom du fichier de configuration avec Windows & my.cnf avec Unix ; il semble que l'on puisse trouver des my.cnf avec Windows, mais je n'en ai jamais vu.

```
The following groups are read: mysqld server mysqld-5.1
The following options may be given as the fiF6ft argument:
--print-defaults      Print the program argument list and exit.
--no-defaults         Don't read default options from any option file.
--defaults-file=#     Only read default options from the given file #.
[...]
```

Comme vous pouvez le constater, paramétrer un SGBD demande de déjà connaître les SGBD. Même l'utilitaire *webmin* qui autorise un paramétrage en mode graphique de MySQL/MariaDB ne dispense de savoir ce que l'on fait. C'est pourquoi nous emploierons *phpMyAdmin* pour commencer : il nous permettra de nous familiariser avec *MySQL/MariaDB* comme avec *PHP* sans pour autant avoir à les paramétrer. Notez que *phpMyAdmin*, n'est pas seulement un client basique, il permet également d'administrer les bases de données.

Le Client

L'usage est de l'invoquer comme ceci : **mysql** -u moi *mabase*, où moi est le nom de l'utilisateur, *mabase* le nom de la base à laquelle il veut se connecter. Mais il existe, là encore, beaucoup d'options ; leur liste est en annexe S05.

LES UTILITAIRES D'ADMINISTRATION

L'administration, à ce niveau, est relative aux bases de données & à leurs utilisateurs. **mysqlmanager** qui disparaîtra dans la version 6 de MySQL permet de créer, de supprimer ou de modifier les droits & les mots de passe des utilisateurs. **mysqladmin** permet de faire la même chose avec les bases de données. L'annexe S05, toujours lui donne la liste exhaustive des commandes disponibles pour ces utilitaires.

LE PILOTE ODBC

Ce pilote ne s'emploie pas avec *MySQL/MariaDB*, mais avec d'autres logiciels, comme *Microsoft SQL-Server* ou *Access* ou encore [*Open*|*Libre*]*Office*[*.org*] *Base*, afin de leur permettre d'accéder à des bases MySQL/MariaDB, non sauvegardées ou actives. *MariaDB* possède un moteur ODBC permettant d'employer des tables en provenance de ces logiciels comme des tables MariaDB.

LES MOTEURS DE BASE DE DONNÉES

Un moteur de base de données (*database engine* ou *storage engine*) est un composant logiciel, intégré dans le SGBD, qui contrôle, écrit, lit & trie des informations dans une ou plusieurs bases de données.

Il existe trois méthodes d'accès à des informations stockées dans une base de données :

- séquentielle, parcours de toutes les informations à partir de la première jusqu'à celle cherchée ;

- directe, accès direct à une information dont on connaît l'emplacement ;
- séquentielle indexée, celle des dictionnaires : organisation des données en pages auxquelles on accède directement puis parcours séquentiel des données dans la page ; l'accès direct de fait au moyen d'un index qui indique position de chaque page.

La méthode employée dans les SGBD relationnels est la dernière, dite *ISAM* (*Indexed Sequential Access Method*). Ce qui a donné le nom du premier moteur de base de données de *MYSQL*, ISAM, rapidement remplacé par MYISAM, plus performant, en voie de remplacement par InnoDB qui respecte les contraintes ACID (cf. p. 34).

Concrètement une base de données est une dossier portant le nom de la base rangé dans le dossier de stockage par défaut du SGBD (par exemple .../mysql.data). Le moteur de base de donnée définit comment seront stockées les tables.

Comme on le verra p. 146, les options ENGINE & TYPE de l'instruction **CREATE TABLE** permettent d'indiquer le moteur de base de données à employer pour accéder aux données d'une table.

MySQL gère les moteurs : BDB, MEMORY, ISAM, InnoDB, MERGE, MRG_MYISAM & MYISAM. *MariaDB* y ajoute : Xtradb amélioration d'InnoDB ; Aria, amélioration de MyISAM ; TokuDB, transactionnel & à fort taux de compression des données ; Cassandra, un moteur NoSQL CONNECT utilisation de toutes sortes de tables, des CSV aux ODBC en passant par les DBF & les XML ; etc.

Petite illustration

Les moteurs influent sur la taille des fichiers & sur la vitesse des opérations, mais comme celle-ci varie en fonction des accès disques sur le serveur & de l'encombrement du réseau, il s'avère difficile de déterminer s'il en est un beaucoup plus rapide que d'autres. Avec

cette table & sur notre serveur, InnoDB est souvent dans les plus rapides. Notez que Aria peut gérer les transactions.

Une table comporte au moins deux fichiers un définissant la structure de la table (extention frm), l'autre les données, certains moteurs (MyISAM, Aria) y ajoute un fichier contenant les index. À titre d'exemple voici comment une même table est stockée, selon le moteur employé.

MOTEUR	TAILLE	NOM
InnoDB	1051	tpers1.frm
	98304	tpers1.ibd
MyISAM	12668	tpers2.MYD
	9216	tpers2.MYI
	1047	tpers2.frm
Aria sans transactions	32768	tpers3.MAD
	16384	tpers3.MAI
	1049	tpers3.frm
Aria avec transactions	32768	tpers4.MAD
	16384	tpers4.MAI
	1049	tpers4.frm
CSV	35	tpers5.CSM
	11803	tpers5.CSV
	555	tpers5.frm

Nous détaillerons le moteur CSV, car ce format de fichier sert au transfert de données depuis & vers les applications bureautiques.

LE MOTEUR CSV

Ce composant, commun à tous les SGBD stocke les données dans au moins un fichier texte pour chaque table. La première ligne contient, éventuellement, le nom des colonnes séparés par un délimiteur (la virgule par défaut) & les lignes suivantes les données (transferts depuis un tableur ou un traitement de texte).

La virgule peut être remplacée par un autre symbole & la définition des colonnes absente.

Exemple 1 :

```
nom;prénom
Pierre;Pierre
Gérard;Gérard
Jean;Jean
```

Vous pouvez dès maintenant suivre le TP 1, qui vous guidera dan,s votre apprentissage de *MySQL* avec *phpMyAdmin*.

Nous allons tenter une première approche des script en *PHP*.

Langages de script & PHP

Nous supposons que vous vous êtes déjà familiarisés avec les scripts, en particulier, à travers notre livre Scripts en Bash, chez le même éditeur (www.le-maitre-refleur.fr) C'est pourquoi nous ne donnerons que de brèves généralités absentes de ce remarquable ouvrage.

Langages de programmations

Un langage de programmation est un langage informatique c'est-à-dire une grammaire & un vocabulaire que l'on peut traduire en langage machine. On appelle *code source* un texte utilisant ce vocabulaire & cette grammaire. Ce texte sera d'abord analysé afin d'en déterminer sa correction orthographique & grammaticale. S'il est correct, il sera traduit par un programme particulier soit directement dans le langage de l'ordinateur sur lequel il doit d'exécuter, soit dans le langage d'une machine virtuelle. Une machine virtuelle [04001] est un programme qui transforme un langage machine virtuel en langage machine réel. En l'absence de machine virtuelle, il faut traduire le programme pour chaque processeur sur lequel on veut l'employer. La machine virtuelle la plus répandue est la machine virtuelle Java de Oracle, ex-SunMicrosystems, elle existe pour quatre SE différents : Solaris avec processeurs SPARC, Solaris, Linux & Windows avec processeurs compatibles Intel.

L'activité de rédaction du code source d'un programme est nommée programmation. Mais avant décrire, il faut savoir ce que l'on va écrire, c'est-à-dire, quel travail le programme devra faire ? & comment il devra le faire ? Pour des travaux complexes, il faut recourir à des méthodes algorithmiques, pour des programmes simples, la maîtrise du français suffira comme nous le verrons.

L'expérience a prouvé que la mise au point d'un programme coûtait plus cher que son écriture. Il y a deux façons de contourner ce problème :

- diviser le programme en morceaux logiques ayant une complexité relativement faible, facilement testables de façon à mettre sur le marché un produit contenant très peu de bugs (optique Unix) ;
- mettre sur le marché un produit bogué & faire payer les clients stupides pour détecter les erreurs (optique Microsoft).

Un langage de programmation repose sur :

- une grammaire, improprement appelée syntaxe [04002] ;
- un alphabet composé de lettres, les minuscules étant parfois considérées comme étant distinctes des majuscules (le Z qui n'est pas la même lettre que le z se place avant le a), de chiffres ;
- de signes de ponctuation variables d'un langage à l'autre (« ; », « , », « | », « _ » etc.) ;
- d'opérateurs (« = », « == », « .eq. », « + », « AND », « & », etc.) ;
- de mots réservés, ce sont des mots dont on ne peut changer le sens (« if », « then », « else », etc.) ;
- de mots clés, ce sont des mots prédéfinis dont on peut ponctuellement changer le sens (« print », « read », « integer », etc.) ;
- de mots prédéfinis ne faisant pas partie des précédents ; il s'agit d'extensions nécessaires pour accomplir des tâches courantes, mais que les concepteurs du langages n'ont pas jugé bons d'intégrer dans celui-ci (« copydir », « changeword », etc.) ;
- de mots créés par le programmeurs pour leurs besoins.

Ces mots sont de quatre sortes :

- *les mots définissant la forme des données* (type de données – **INTEGER**, **VARCHAR**, etc.–, classe d'objets) ;
- *les données* (variables –*`table`.`colonne`*–, constantes –*732*, 'AZERTY', etc. –, objets) ;

- les mots instructions, équivalents des verbes d'une langue vernaculaire (**INSERT**, **CREATE**, **DROP**, **SELECT**, etc.) ;
- *les fonctions* mélanges d'instruction & de données, puisqu'elles ont une action comme les instructions & une valeur comme les données.

L'assemblage de ces éléments permet de former des phrases que l'on appelle des instructions. Ces instructions sont de cinq sortes différentes :

- *séquentielles*, elles s'exécutent à la suite les une des autres ;
- *conditionnelles*, l'exécution d'une action étant soumise à la véracité d'une condition ;
- *itératives*, les instructions sont répétées autant de fois que nécessaire ;
- *de groupement*, les instructions sont solidaires les unes des autres ;
- *fonctionnelles*, elles sont alors créées directement par le programmeurs ou elles ont été créées par d'autres programmeurs.

Mais tous les langages ne sont pas algorithmiques, d'autres, comme le SQL ou le HTML sont dit déclaratifs. Cela revient à dire qu'au lieu de dérouler une séquence d'instruction, on écrit une liste de déclarations par exemple *ceci est le titre en couleur rouge* (*HTML* : `<h1 style="color:#ba0002">Ceci</h1>`) ou *créer la table réalisateur dans la base dvd* (*SQL* : `CREATE TABLE `dvd`.`realisateur``) & le programme traducteur, beaucoup plus complexe que le précédent, effectue la traduction & exécute le travail demandé.

Programmation

Écrire un programme c'est un peu élaborer une recette de cuisine.

Dans une recette de cuisine, on a une situation de départ (des ingrédients des ustensiles, un appareil de cuisson & un cuisinier) & une situation d'arrivée, le plat à réaliser que l'on doit faire dans le nombre fini d'étapes décrit par la recette. Certaines étapes sont

séquentielles, d'autres sont itératives (il vous faudra découper tous les ingrédients prévus) d'autres sont alternatives (si le plat est cuit avant le temps prévu arrêter la cuisson).

La programmation déclarative en cuisine est rarissime : seuls, les grand chefs, qui ont une escouade de cuisiniers peuvent s'y adonner !

Il y a deux façons de traduire d'une langue dans une autre : sur le coup avec un interprète ; après coup avec un traducteur. Dans le premier cas le programme employé est dénommé interpréteur, dans le second, compilateur. Les derniers fournissent une bien meilleure traduction que les premiers, mais ils sont moins conviviaux. Aujourd'hui avec les progrès des processeurs & les quantités de mémoire disponibles, malgré leur lenteur, les programmes interprétés deviennent acceptables (leurs temps de réponse n'irritent pas l'utilisateur). De plus, le développement d'applications fonctionnant via Internet réduit ce handicap, puis que si lent que soit un programme interprété, il va toujours plus vite, en 2015, qu'une liaison haut débit.

Il existe des milliers de langages de programmation ; la plupart d'entre eux sont réservés à des domaines spécialisés.

Certains ont connus, pour des raisons rarement en rapport avec leurs qualités, un grand succès, comme le BASIC ou le C. Les quatre raisons principales du succès d'un langage sont la force marketing de son éditeur, la volonté de l'imposer de son créateur, le chauvinisme américain, la réponse à un besoin. Si, en plus, c'est un produit de qualité, tout est parfait.

Chaque langage a ses caractéristiques propres.

PARTICULARITÉS

Il existe plusieurs caractéristiques qui permettent de comparer les langages entre-eux.

* Le *typage* est le fait d'attribuer un type aux éléments du code source (variables, fonctions, etc.). Un type définit une structure de

données : nombre entier, tableau, chaîne de caractères. Dans certains langages, il existe des types plus évolués (liste chaînée, sémaphore) & il est parfois possible d'en définir de nouveaux.

Les types que l'on trouve dans un langage de programmation dépendent de sa sémantique &, donc, de ses paradigmes. Beaucoup de langages proposent la notion de *variable* qui associe un nom à une valeur en mémoire & ce nom ou cette valeur à un type.

- Le typage peut être explicite ou implicite.
  * On parle de typage explicite quand les types apparaissent explicitement dans le code source du programme ;

Exemple 2 : en C

```
unsigned int compteur ; // déclare compteur comme une variable entière dont les valeurs sont comprises entre 0 & 65535
char phrase_courte ; déclare phrase_courte comme étant une chaine de 0 à 255 caractères terminée par le caractère de code ASCII 0
float CA_mens[12] ; // définit un tableau de 12 nombres décimaux repérés de 0 à 11.
```

  * un typage implicite est déterminé par le compilateur ou l'interprète.

Exemple 3 : en PHP

```
1 compteur=100 //l'interpréteur en déduit que compteur est un nombre entier.
```

- Il peut également être *fort* ou *faible*. Plus un typage est fort, plus les règles du langage sont strictes & interdisent, dès la compilation, les manipulations entre données de types différents. On trouve souvent dans les langages typés faiblement la possibilité de faire du *transtypage* manuel (en anglais *cast*) pour combler les lacunes du système automatique de typage.
- Ou encore *statique* ou *dynamique*. On parle de typage statique quand la vérification des types se fait dès la phase de compilation, & de typage dynamique Lorsque celle-ci est faite durant

l'exécution. De plus un typage dynamique associe les types aux valeurs, alors qu'un typage statique associe le type à la variable.

Le typage du langage C est explicite, relativement fort (le compilateur peut générer des avertissements de typages, le transtypage manuel est permis, mais on peut effectuer n'importe quelle opération entre n'importe quels types sans transtypage), & statique. Le langage Objective Caml possède un typage implicite, fort & statique.

Le typage du SQL est fort & statique, celui du PHP, faible & dynamique.

Les langages fournissent parfois des mécanismes pour convertir une valeur d'un type en une valeur dans un autre type : on peut convertir un entier en décimal sans aucune perte mais l'inverse n'est pas toujours possible [04003].

* Outre les particularités sémantiques des langages, ils utilisent également des grammaires différentes qui proposent souvent des points communs.

Les particularités syntaxiques ne sont souvent que des détails qui ne changent pas les fonctionnalités proposées par les langages de programmation. Par exemple, dans Objective-C 2, il est possible d'utiliser les accesseurs (mot du langage servant à récupérer une information) avec deux syntaxes différentes pour le même résultat.

- Les commentaires sont des parties du programme qui n'apparaissent pas dans l'application finale. Les commentaires permettent de documenter & d'expliquer le code source. Presque tous les langages de programmation permettent d'écrire des commentaires.

Ils sont introduits ou délimités par des caractères spéciaux (# en Bash ou Ruby, // ou /* en C, en Perl ou en PHP) ou une instruction particulière (REM en BASIC).

- L'*indentation* est l'utilisation d'espaces ou de tabulations en début de ligne. Généralement, elle n'a pas d'incidence sur le

fonctionnement du programme & ne sert qu'à améliorer la lisibilité du code.

Pour certains langages, l'indentation est significative & obligatoire : en Python, l'indentation va servir à délimiter une *fonction*, une *classe* ou un *test conditionnel*, à la différence de langages comme C où ce sont les accolades, { & }, qui remplissent cette fonction.

- Pour distinguer une instruction de la suivante, il existe principalement deux approches :
  * soit la fin d'une instruction est marquée par un terminateur (un ; en C, Java, etc.) ;
  * soit il existe un séparateur d'instructions (un ; en Pascal, une fin de ligne en Bash ou Python).

La nuance est importante, car en C, la dernière instruction d'un bloc doit comprendre un ; (terminateur) alors qu'en Pascal, il est inutile, voire fautif, d'en mettre un (cela consiste à ajouter une instruction vide en fin de bloc). De même, le caractère ; peut être utilisé en C comme instruction vide, ce qui n'aurait pas de sens en Pascal.

Les langages utilisant un terminateur sont réputés engendrer moins de fautes de syntaxe que ceux utilisant un séparateur (Le C étant le contre-exemple !)

- Certains langages utilisent des *balises* délimitant des *environnements*. Une balise est généralement un *mot-clé* associé à plusieurs caractères : \ ; < > ; etc.

Les langages permettant de générer du code source ou des documents utilisent souvent des balises. Par exemple, PHP & JSP utilisent des balises pour délimiter les parties de code à interpréter. XML est également un langage à balises, qui permet de définir des langages de programmation comme XSP ou XSLT.

- Une stratégie d'évaluation est un ensemble de règles qui décrivent comment évaluer une expression dans un langage de

programmation. La stratégie d'évaluation définit à quel moment les arguments des fonctions & opérateurs sont évalués ou réduits. On distingue essentiellement deux stratégies :

* l'évaluation stricte : les arguments des fonctions sont toujours évalués avant que la fonction ne soit appliquée.
* l'évaluation paresseuse ou évaluation retardée : les arguments ne sont évalués que lorsque leur valeur est effectivement requise. Ce type d'évaluation est généralement utilisée dans les langages fonctionnels.

La plupart des langages ont un système d'évaluation stricte, & utilisent une forme d'évaluation paresseuse pour les expressions booléennes (évaluation court-circuit). Cependant, il est possible de créer une évaluation paresseuse dans un langage à évaluation stricte. Par exemple, Scheme fournit la fonction delay qui retarde l'évaluation d'une expression & force qui en oblige l'évaluation.

⁂ Les langages de programmation offrent plus ou moins de libertés au programmeur en ce qui concerne la gestion de la mémoire.

- Soit elle est entièrement sous le contrôle du développeur qui doit gérer lui-même l'espace mémoire disponible, les allocations & libérations (C).
- Soit tout peut être contrôlé par le compilateur ou par le moteur d'exécution (Java) : même s'il est possible de donner des directives, la machine virtuelle gère elle-même la mémoire à l'aide d'un ramasse-miettes.

Certains langages proposent un système intermédiaire. En Objective-C, il est possible de la gérer directement, d'activer la gestion automatique de la mémoire ou d'utiliser un système de plus haut niveau.

⁂ On dit d'un langage de programmation est *réflexif* s'il permet, au moment de l'exécution, d'analyser & d'agir sur le fonctionnement interne du programme lui-même.

Le langage Smalltalk fut un précurseur dans le domaine de la réflexivité. Il a d'ailleurs fortement influencé bon nombre de langages réflexifs, tels que Python, Ruby ou Java.

⁕ Les exceptions sont des cas limites d'exécution du programme (division par 0, etc.) Lorsqu'elles sont générées, le déroulement du programme est interrompu. Au contraire d'une erreur qui interrompt brutalement le programme, la gestion d'une exception permet d'interrompre proprement un programme, de corriger l'erreur & d'en reprendre l'exécution.

La gestion des exceptions peut être différente selon les langages soit :

- il n'y en a pas (C) ;
- elle signale l'exception sans la traiter (C++, Java) ;
- elle permet un traitement : par exemple, la modification du programme par le programme lui-même (comme en Python) pour reprendre l'exécution *normalement*.

⁕ La programmation concurrente consiste à découper un programme en plusieurs fils d'exécution. La concurrence permet de simuler l'exécution de différentes tâches de façon simultanée ou, dans le cadre d'une interface graphique, d'effectuer des tâches en tâche de fond sans pour autant bloquer le rafraîchissement de l'affichage. La programmation concurrente permet aussi de tirer parti de ressources distribuées (multiples processeurs, cluster de machines, etc.) ou du système d'ordonnancement de processus du système d'exploitation.

Certains langages intègrent directement la concurrence dans leurs primitives (Erlang, Concurrent ML). Généralement, la concurrence est intégrée par l'intermédiaire de bibliothèques spécifiques : le langage C dispose entre autres de la bibliothèque des *threads* POSIX ; Java dispose d'une classe **Thread** dans ses bibliothèques standards.

⁕ Chaque langage de programmation est différent. Une solution exprimée dans un certain langage peut *ressembler* à une solution

exprimée dans un autre langage ; dans ce cas, on dit que les langages utilisent le même paradigme (ou style). Deux programmes fournissant la solution au même problème, mais écrits avec des paradigmes différents seront fondamentalement très différents.

De nombreux langages appartiennent simultanément à plusieurs catégories : ils sont dits *multi-paradigmes*. Par exemple, C++ permet la *programmation impérative*, *orientée objet* & la *programmation générique* (à base de classes & de fonctions paramétrées nommées *templates*). Common Lisp est à la fois *impératif*, *fonctionnel*, *orienté objet* ; son caractère *évolutif* lui permet d'intégrer d'autres paradigmes de programmation en son sein (par exemple : la programmation logique & la programmation par contraintes).

- On distingue deux types de langages impératifs.
  - D'une part les langages machines & assembleurs. Ceci explique que les premiers langages de programmation apparus soient des langages impératifs : une instruction du langage correspond à un ensemble d'instructions du langage machine. Les structures de données & opérations sont plus complexes qu'au niveau de la machine, mais le paradigme suivi reste le même.
  - Les langages procéduraux forment la seconde famille de langages impératifs. Une procédure, appelée également fonction, est une suite d'instructions devant être effectuée dans un ordre précis. On distingue parfois procédure & fonction par la caractéristique qu'une procédure ne renvoie pas de résultat.

  Parmi les langages impératifs figurent COBOL, Fortran, Pascal ou encore le C.

Un langage déclaratif ne décrit pas comment est réalisée une opération, comme dans un langage impératif, mais décrit le problème lui-même, sans s'intéresser au contexte.

Figurent parmi les langages déclaratifs Oz, Prolog ou encore Clips.

⁕ Un programme logique est composé de faits & de règles qui sont traités par un moteur d'inférence.

Prolog fut le premier langage de ce type à être opérationnel.

⁕ Dans ce paradigme, l'opération de base n'est pas l'affectation, contrairement aux langages impératifs, mais l'évaluation de fonctions. Ce paradigme est donc principalement efficace pour modéliser des problèmes qui s'expriment par des valeurs de données, comme en mathématiques, & non pas des états qui changent au cours de l'exécution.

Certains langages fonctionnels, dit *purs*, interdisent totalement les effets de bord, tels que la mutation des données où une variable est liée à une valeur non modifiable. D'autres, intègrent certains traits des langages impératifs où les variables peuvent changer de valeur au cours de l'exécution.

Quelques exemples de langages fonctionnels : Objective Caml (langage fonctionnel & impératif), Haskell (langage fonctionnel pur à évaluation paresseuse), Python (propose certaines fonctionnalités du style fonctionnel comme l'utilisation de fonctions lambda ou la compréhension de listes).

⁕ Ces langages ne manipulent que des piles de données où les opérations sont effectuées sur les éléments du sommet d'une ou plusieurs piles.

Des exemples typiques : les langages Forth, PostScript ou RPL (HP 48).

⁕ Les langages à objets offrent une abstraction à la machine : l'objet est une structure sémantique indépendante qui rassemble des données & des traitements.

En se basant sur une méthode de conception à objet & sur un langage de modélisation à objet, on peut facilement implanter un concept au moyen d'un langage de programmation à objets.

Parmi les langages à objets sont généralement classés Ruby & Smalltalk, purement objet, c'est-à-dire que tout est objet, depuis les

types de base, jusqu'à l'interprète ou les blocs d'instructions ; le C++, extension de C permettant d'utiliser des objets mais où tout n'est pas objet ; Python, très orienté objet mais qui ne respecte pas tous les principes de la programmation objet comme l'encapsulation.

UTILISATIONS

On peut aussi classer les langages de programmation en fonction de leur utilisation car beaucoup de langages sont spécialisés à une application ou à un domaine particulier.

* Un langage de définition de données ne permet pas d'effectuer de traitement mais de décrire des structure de données (listes, arbres...) & des instances de ces structures.

  XML est par exemple un langage permettant la représentation de données sous forme de structure arborescente ; la partie DDL de SQL sert à décrire des données relationnelles.

* Les langages qui décrivent des documents peuvent également être considérés comme des langages de définition de données. Ainsi, LaTeX est un exemple de langage de définition de données qui permet d'écrire un document en centralisant sa mise en forme. LaTeX est, ensuite, compilé vers d'autres langages de description de documents, généralement de plus *bas niveau*, comme le Postscript.

  Ces langages ne sont habituellement pas considérés comme des langages de programmation. Cependant les codes sources produits avec ces langages présentent certains traits de codes sources de programmes comme des structures de contrôle (conditions, boucles...) & des moyens d'interaction avec le système (variable d'environnement, formulaires...). Ils sont donc cités ici à titre indicatif, mais ils sont considérés comme à la frontière de la programmation, par les codeurs fous.

* Les langages de requêtes sont destinés à interroger & manipuler les bases de données.

SQL est un langage de requête utilisé par un grand nombre de systèmes de gestion de bases de données tels que *MySQL*, *Oracle* ou *SQL Server*. Les SGBD dit *NoSQL* (*Not only SQL*) emploie, en plus du SQL, un autre langage de recherche.

⁕ Les langages de script sont utilisés soit pour une plus grande interaction entre un client & un serveur, soit pour administrer un logiciel ou un système d'exploitation [04004].

Du côté du serveur web, cela permet de produire des pages dont le contenu est généré à chaque affichage. Ces langages sont par ailleurs souvent couplés avec un langage pour communiquer avec des bases de données (exemple : PHP).

Côté client (en général le navigateur web), les langages peuvent réagir à certaines actions de l'utilisateur sans avoir à questionner le serveur. Par exemple, le JavaScript d'une page web peut répondre aux saisies de l'utilisateur dans un formulaire, afin de vérifier le format des données.

Certains langages permettent de développer à la fois les aspects client & serveur. C'est le cas d'Ocsigen, de Hop ou bien encore du Server-Side JavaScript.

⁕ On désigne, parfois, par *langages de programmation théoriques*, les *systèmes formels utilisés pour décrire* de façon théorique le fonctionnement des ordinateurs. Ils ne servent pas à développer des applications mais à représenter des modèles & démontrer certaines de leurs propriétés.

On peut citer la machine de Turing & le λ-calcul de Church, qui datent tous les deux des années 1930, & donc antérieurs à l'invention de l'ordinateur. Le λ-calcul a, par la suite, servi de base théorique à la famille des langages de programmation fonctionnelle. Dans les années 1980, Robin Milner a mis au point le π-calcul pour modéliser les systèmes concurrents.

⁕ Les langages exotiques ont pour but de créer des grammaires complètes & fonctionnelles mais dans un paradigme éloigné des conventions. Beaucoup sont d'ailleurs considérés comme des blagues.

Ces langages sont généralement difficiles à mettre en pratique & donc rarement utilisés. Par exemple, le Piet permet de programmer à l'aide d'images matricielles.

On peut également citer le Brainfuck qui est un *langage minimaliste & TURING-complet* (8 instructions seulement). Il est prévu pour tourner sur une machine de TURING avec un compilateur de seulement 171 octets.

Il existe également des langages spécialisés :

- ABEL, langage pour la programmation électronique des circuits logiques programmables (*PLD* en jargon) ;
- CDuce, langage fonctionnel d'ordre supérieur pour la manipulation de documents au format XML ;
- Forme de Backus-Naur (BNF), formalisation des langages de programmation ;
- PROMELA, langage de spécification de systèmes asynchrones ;
- VRML, description de scènes en trois dimensions.

⁕ Langages de programmation de *Commande Numérique* (CN)

Une machine-outil automatisée, ou *Commande Numérique* , a besoin d'un langage de programmation pour réaliser les opérations de tournage, ou de fraisage...

Sequential function chart est un langage graphique, dérivé du grafcet (NB : le grafcet définit les spécifications de ces machines de façon graphique).

Langages de scripts

Larry Wall, qui est le concepteur du langage de programmation Perl, a dit :

- *When I was a SRTS programmer on a PDP-11, I certainly treated BASIC as a scripting language, at least in terms of rapid prototyping and process control. I'm sure it warped my brain forever...* (*Quand je programmais sous SRTS sur un PDP-11, j'ai assurément traité BASIC comme un langage de script, en tout cas pour le prototypage & la commande de processus. Je suis certain que cela m'a déformé durablement*).
- *Basically, scripting is not a technical term. When we call something a scripting language, we're primarily making a linguistic and cultural judgment, not a technical judgment.*(*Langage de script ne constitue pas un terme technique; l'utiliser fait état d'un jugement linguistique & culturel, pas technique*)

Un langage de script est un langage de programmation qui permet de manipuler les fonctionnalités d'un système informatique configuré pour fournir à l'interpréteur de ce langage un environnement & une interface qui déterminent les possibilités de celui-ci. Le langage de script peut alors s'affranchir des contraintes de bas niveau (prises en charge par l'intermédiaire de l'interface) & bénéficier d'une syntaxe de haut niveau.

Le langage de script est généralement exécuté à partir de fichiers contenant le code source du programme qui sera interprété. Historiquement, ils ont été créés pour raccourcir le processus traditionnel de développement édition-compilation-édition des liens-exécution propre aux langages compilés. Les premiers langages étaient souvent appelés *langage de commande* ou *langage d'enchaînement des travaux* (*JCL : Job Control Language*), car ils permettaient simplement d'automatiser une succession de commandes simples, à la

manière d'un *script* de théâtre. Par la suite, ils furent munis d'exécutions conditionnelles implicites (IBM 1130) ou explicites (JCL), & enfin d'ordres de boucle & d'opérateurs les transformant en quasi-langages de programmation.Dans le sens le plus traditionnel, qui est celui des *shell scripts*, *un* script *sert principalement à lancer & coordonner l'exécution de programmes*.

Dans sa version la plus simple, un script ne spécifie qu'une suite de programmes à appeler dans un ordre donné (par exemple éditeur de texte, compilateur, éditeur de liens & exécution du code objet). Cela crée rapidement des inefficiences : à quoi bon en effet tenter de charger & d'exécuter un programme dont la compilation a échoué ?

On y ajoute donc assez vite une possibilité d'exécution conditionnelle simple (||, &&) en fonction du résultat de l'étape immédiatement précédente (il ne sert à rien d'exécuter si la compilation est mauvaise).

La possibilité d'y employer des variables, des paramètres, des structures de contrôle (répétition, exécution conditionnelle), etc. fait des langages de scripts de véritables langages de programmation.

Dans un sens différent, on appelle aussi langage de script, un langage où les éléments visuels sont considérés comme des personnages placés sur une *scène*, personnages dont le comportement est défini par un *script*. L'un des premiers langages de ce type dans le monde de la micro-informatique a été le langage HyperTalk (langage orienté objet dérivé de Smalltalk) très proche du langage naturel, que DAN WINKLER créé en 1987, pour commander le programme HyperCard sur MacIntosh. Le langage Lingo de Macromedia Director est un descendant direct d'HyperTalk.

Enfin, l'informatique n'étant pas à une terminologie approximative près, *langage de script* désigne parfois, dans un sens très vague, n'importe quel langage de programmation interprété (ce qui est un abus de langage : en toute rigueur le caractère interprété ou compilé se rapporte à une implantation particulière, pas à un langage),

par rapport aux langages toujours compilés comme C, C++, Java, etc. On y trouve alors ceux qui sont parfois ou toujours interprétés comme BASIC, PHP, Lisp, JavaScript, etc.

Le terme langage de script a souvent une connotation négative, on préfère alors parler de *langage de programmation dynamique* quand c'est possible.

Les langages de script les plus connus sont :

- sh ; bash ; ksh ; zsh ; csh ; tcsh ;
- JavaScript ;
- AppleScript (peut être compilé) ;
- VBScript ;
- Lisp ; Scheme (peuvent être compilés) ;
- PHP ;
- Perl (peut être compilé) ;
- Python (peut être compilé) ;
- Ruby ;
- Lua ;
- etc.

Mais il en existe beaucoup d'autres.

Les scripts sont utilisés à différents niveaux :

- côté OS comme une méthode d'automatisation (VBA de Microsoft, Python de *Blender*) ;
- côté serveur web, dits *Server Side Includes*, comme une méthode pour générer des pages dynamiques (JSP, PHP, ASP, ASP.NET) ;
- côté navigateur, dits *Client Side*, sous forme d'*applet* (actionsscripts C#, applet java) ou d'automatisation (VBA).

C'est (presque) fini pour la partie culturelle ! Nous allons maintenant nous consacrer aux langages PHP & SQL.

PHP

Cette section reprend la page Wikipédia consacrée à PHP (*PHP : Hypertext Preprocessor*) & le manuel PHP. Celui-ci est un langage de scripts, libre, principalement utilisé pour produire des pages web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale, en exécutant les programmes en ligne de commande. PHP est un langage impératif disposant depuis la version 5 de fonctionnalités de modèle objet complètes. En raison de la richesse de sa bibliothèque, on désigne parfois PHP comme une plate-forme plus qu'un simple langage.

Historique

Ce langage fut créé en 1994 par Rasmus Lerdorf pour son site web. C'était à l'origine une bibliothèque logicielle en Perl dont il se servait pour conserver une trace des visiteurs qui venaient consulter son CV. Au fur & à mesure qu'il ajoutait de nouvelles fonctionnalités, Rasmus a transformé la bibliothèque en une implantation en langage C, capable de communiquer avec des bases de données & de créer des applications dynamiques & simples pour le web. Rasmus décida alors en 1995 de publier son code, pour que tout le monde puisse l'utiliser & en profiter. PHP s'appelait alors PHP/FI (pour *Personal Home Page tools/Form Interpreter*). En 1997, deux étudiants, Andi Gutmans & Zeev Suraski, redéveloppèrent le cœur de PHP/FI. Ce travail aboutit un an plus tard à la version 3 de PHP, devenu alors PHP: Hypertext Preprocessor. Peu de temps après, Andi Gutmans & Zeev Suraski commencèrent la réécriture du moteur interne de PHP. Ce fut ce nouveau moteur, appelé *Zend Engine*, qui servit de base à la version 4 de PHP.

En 2002, PHP était utilisé par plus de 8 millions de sites web à travers le monde, en 2007, par plus de 20 millions & en 2013, par 244 millions.

Un indicateur paradoxal de la popularité de PHP est le nombre de failles de sécurité concernant des applications PHP & listées avec un identifiant CVE sur la National Vulnerability Database, base de données américaine. Ces failles représentent 12 % du total en 2003, 20 % en 2004, 28 % en 2005, 43 % en 2006, 36 % en 2007, 38 % sur les deux premiers mois de 2008. Plus d'un quart des vulnérabilités répertoriées sur cette base concerne des applications PHP, plus d'un tiers ces dernières années, & la plupart peuvent être exploitées à distance. Ces vulnérabilités s'expliquent par de mauvaises habitudes de programmation (souvent un défaut de validation des entrées) alliées à des caractéristiques douteuses du langage lui-même (par exemple register globals, maintenant déconseillé).

La version actuelle est la version 5, sortie le 13 juillet 2004. Elle utilise Zend Engine 2 & introduit un véritable modèle objet, une gestion des erreurs fondée sur le modèle des exceptions, ainsi que des fonctionnalités de gestion pour les entreprises. PHP 5 apporte beaucoup de nouveautés, telles que le support de *SQLite*, qui est un système léger de gestion de bases de données embarqué, au détriment de la bibliothèque cliente de MySQL/MariaDB, plus puissante mais qui n'est désormais plus activée par défaut, ainsi que des moyens de manipuler des fichiers & des structures XML basés sur *libxml2* :

- une API simple nommée *SimpleXML* ;
- une API Document Object Model assez complète ;
- une interface *XPath* utilisant les objets *DOM* & *SimpleXML* ;
- une intégration de *libxslt* pour les transformations *XSLT* via l'extension *XSL* ;
- une bien meilleure gestion des objets par rapport à PHP 4, avec des possibilités qui tendent à se rapprocher de celles de Java.

La dernière mise à jour est la 5.6 (avril 2015).

PRÉSENTATION

Ce langage est utilisé principalement en tant que langage de script côté serveur, ce qui veut dire que c'est le serveur (la machine qui héberge la page web en question) qui va interpréter le code PHP & générer du code (constitué généralement d'XHTML ou d'HTML, de CSS, & parfois de JavaScript) qui pourra être interprété par un navigateur. PHP peut également générer d'autres formats en rapport avec le web, comme le WML, le SVG, le format PDF, ou encore des images bitmap telles que JPEG, GIF ou PNG.

Il a été conçu pour permettre la création d'applications dynamiques, le plus souvent dédiées au web. PHP est très majoritairement installé sur un serveur *Apache*, mais peut être installé sur les autres principaux serveurs HTTP du marché, par exemple IIS. Ce couplage permet de récupérer des informations issues d'une base de données, d'un système de fichiers (contenu de fichiers & de l'arborescence) ou plus simplement des données envoyées par le navigateur afin d'être interprétées ou stockées pour une utilisation ultérieure.

C'est un langage peu typé & souple & donc facile à apprendre par un débutant mais, de ce fait, des failles de sécurité peuvent rapidement apparaître dans les applications. Pragmatique, PHP ne s'encombre pas de théorie & a tendance à choisir le chemin le plus direct. Néanmoins, le nom des fonctions (ainsi que le passage des arguments) ne respecte pas toujours une logique uniforme, ce qui peut être préjudiciable à l'apprentissage.

Une fois la syntaxe des structures de base acquise, son apprentissage se poursuit par le traitement des formulaires, puis par l'accès aux bases de données. L'accès aux bases de données est aisé une fois l'installation des modules correspondant effectuée sur le serveur. La force la plus évidente de ce langage est qu'il a permis au fil du temps la réalisation aisée de problèmes autrefois compliqués & est devenu par conséquent un composant incontournable des offres d'hébergements. Nous y intercalerons des scripts d'administration système, car il semble plus facile d'emploi que *bash* pour ces travaux.

Il est multi plate-forme : autant avec Linux qu'avec Windows il permet aisément de reconduire le même code sur un environnement à peu près semblable (à l'arborescence des dossiers près).

Libre, gratuit, simple d'utilisation & d'installation, ce langage nécessite comme tout langage de programmation une bonne compréhension des principales fonctions usuelles ainsi qu'une connaissance aiguë des problèmes de sécurité liés à ce langage.

La version 5.3 a introduit de nombreuses fonctionnalités : les *espaces de noms* (un élément fondamental de l'élaboration d'extensions, de bibliothèques & de *frameworks* structurés), les *fonctions anonymes*, les *fermetures*, etc.

La version 7 introduira en interne la bibliothèque ICU donnant au langage la faculté de traiter Unicode de manière native.

Fonctionnement

PHP appartient à la grande famille des descendants du C, dont la syntaxe est très proche. En particulier, sa syntaxe & sa construction ressemblent à celles des langages Java & Perl, à la différence que du code PHP peut facilement être mélangé avec du code HTML au sein d'un fichier PHP.

Dans une utilisation web, l'exécution du code PHP se déroule ainsi : lorsqu'un visiteur demande à consulter une page web, son navigateur envoie une requête au serveur HTTP correspondant. Si la page est identifiée comme un script PHP (généralement grâce à l'extension .php), le serveur appelle l'interprète PHP qui va traiter & générer le code final de la page (constitué généralement d'HTML ou de XHTML, mais aussi souvent de CSS & de JS). Ce contenu est renvoyé au serveur HTTP, qui l'envoie finalement au client.

Ce schéma explique ce fonctionnement :

Une étape supplémentaire est souvent ajoutée : celle du dialogue entre PHP & la base de données. Classiquement, PHP ouvre

une connexion au serveur de SGBD voulu, lui transmet des requêtes & en récupère le résultat, avant de fermer la connexion.

Image 1: Traitement d'un script interprété par PHP

L'utilisation de *PHP* en tant que générateur de pages web dynamiques est la plus répandue, mais il peut aussi être utilisé comme langage de programmation en ligne de commande sans utiliser de serveur HTTP ni de navigateur.

Pour réaliser un script PHP exécutable en ligne de commande, il suffit comme en Perl ou en Bash d'insérer dans le code en première ligne le *shabang* : #!/usr/bin/php (/usr/bin/ est le répertoire standard des fichiers binaires exécutables sur la plupart des distributions).

Il existe aussi une extension appelée PHP-GTK permettant de créer des applications clientes graphiques sur un ordinateur disposant de la bibliothèque graphique GTK+, ou encore son alternative WinBinder.

PHP possède un grand nombre de fonctions permettant des opérations sur le système de fichiers, la gestion des bases de données, des fonctions de tri & hachage, le traitement de chaînes de caractères, la génération & la modification d'images, des algorithmes de compression...

Le moteur de Wikipédia est écrit en PHP avec une base MariaDB. Ce langage s'intègre dans une plate-forme LAMP [05001].

SYNTAXE

Pour commencer, quelques exemples du traditionnel Hello world, premier programme de toutes les initiations respectables à un langage de programmation.

Exemple 4 : echo & printf

```php
<?php
    // forme la plus simple, recommandée
    echo 'Hello World';

    // équivalent à la fonction printf() du langage C
    printf('Hello %s', 'World');
?>
```

echo étant une structure du langage, il est possible (& même recommandé) de ne pas mettre de parenthèses, ce qui lui permet en plus d'accepter plusieurs paramètres.

Exemple 5 : echo sans parenthèses

```php
<?php
    echo 'Hello ', 'World';
?>
```

Résultat affiché :

```
Hello World
```

BALISES

Le code PHP doit être inséré entre des balises <?php & ?>.

Il existe des notations raccourcies : <? & ?>, ou la notation ASP <% & %>, mais celles-ci sont déconseillées, car elles peuvent être désactivées dans la configuration du serveur : la portabilité du code est ainsi réduite. De plus, *PHP 6* (ou *7*) interdira ces notations, ce qui compromettra la compatibilité des scripts utilisant ces balises avec les futures versions de *PHP*.

Il existe aussi cette syntaxe, peu courante : <script language="php"> & </script>.

Constantes, variables & types

Les données manipulées sont soient constantes, soient variables & elles sont typées, mais il s'agit d'un typage implicite (c'est la donnée qui détermine le type) & fluctuant (le type de la variable change avec sa valeur).

Pour obtenir des informations sur une variable, nous disposons pour l'instant de quatre moyens : les instructions echo/print & les fonctions var_dump(), print_r() & var_export().

Exemple 6 : fluctuation du type & outils d'information sur une variable

```
1   #!/usr/bin/php
2   <?php
3   $a=true;
4   echo $a, "-->";
5   var_dump($a);
6   print_r($a);
7   var_export($a);
8   echo "\n";
9   $a=245;
10  echo $a, "-->";
11  var_dump($a);
12  print_r($a);
13  var_export($a);
14  ?>
```

Résultat

```
1-->bool(true)
1true
245-->int(245)
245245
```

Si les deux dernières fonctions sont peu usitées, la fonction var_dump() s'avère pratique pendant la phase de débogage, particulièrement pour les variables complexes.

Il est possible de nommer des constantes telles les taux de TVA. Deux moyens permettent d'y parvenir le mot clé const & la fonction define().

```
1  <?php
2    const TVA_NORMALE= 0.20;
3    define(TVA_RÉDUITE, 0.55);
4    echo 'taux de TVA normale = '.(TVA_NORMALE*100). ' %';
5    echo 'taux de TVA réduite = '.(TVA_RÉDUITE*100). ' %';
6  ?>
```

L'usage est d'écrire les constantes nommées en majuscules.

Il existe des constantes prédéfinies. TRUE & FALSE en sont deux exemples, ce sont des mots qui ne sont pas sensibles à la casse. Ce n'est pas le cas des autres noms de constante comme PHP_EOL (fin de ligne), PHP_VERSION (version de l'interpréteur), etc. Il existe en outre des constantes dites magiques, car leur valeur change en fonction du contexte, mais reste constant dans le contexte : __FILE__ & __DIR__ contiennent respectivement le nom complet du fichier & celui du dossier dans lequel il se trouve, ces valeurs sont constantes dans le script ; __FUNCTION__ qui indique le nom de la fonction en cours d'exécution est constant dans la fonction, etc.

Variables

En PHP, les variables sont représentées par un signe dollar, $, suivi du nom de la variable. Le nom est sensible à la casse.

Les noms de variables suivent les mêmes règles de nommage que les autres entités PHP. Un nom de variable valide doit commencer

par une lettre ou un souligné, _, suivi de lettres, chiffres ou soulignés. Exprimé sous la forme d'une expression régulière, cela donne : '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'.

Note : Dans nos propos, une lettre peut être a-z, A-Z, & les octets de 127 à 255 (*0x7f-0xff*). Bien entendu, on ne tape pas le code hexadécimal du caractère (é & non *0xe9*).

Note : ***$this*** est une variable spéciale, qui ne peut pas être assignée, c'est l'interpréteur qui détermine sa valeur.

Exemple 7 : Validité des noms de variables

```
1  <?php
2  $var = 'Jean';
3  $Var = 'Paul';
4  echo "$var, $Var";          // affiche "Jean, Paul"
5  $4site = 'pas encore';      // invalide : commence par un
   nombre
6  $_4site = 'pas encore';     // valide : commence par un
   souligné
7  $pêche = 'nectarine';       // valide : 'ê' a le code ASCII
   étendu 234 (OxEA)
8  $p0xeache = 'brugnon';      // valide : 0xea n'est pas un nombre
   hexa décimal mais une suite de caractères
9  ?>
```

Par défaut, les variables sont assignées par valeur : lorsque vous assignez une expression à une variable, la valeur de l'expression est recopiée dans la variable. Cela signifie, par exemple, qu'après avoir assigné la valeur d'une variable à une autre, modifier l'une des variables n'aura pas d'effet sur l'autre.

PHP permet aussi d'assigner les valeurs aux variables par référence. Cela signifie que la nouvelle variable ne fait que référencer (en d'autres termes, *devient un alias de*, ou encore *pointe sur*) la

variable originale. Les modifications de la nouvelle variable affecteront l'ancienne & vice versa.

Pour assigner par référence, ajoutez simplement un & (esperluette) au début de la variable qui est assignée (la variable source). Dans l'exemple suivant, Mon nom est Pierre s'affichera deux fois.

Une chose importante à noter est que seules les variables définies dans le programme peuvent être assignées par référence.

Une assignation par référence revient à lier deux variables, chaque modification de la variable référencée est automatiquement copiée dans la zone mémoire de la variable *référençante*. Une nouvelle assignation de la variable *référençante* supprime le lien.

Exemple 8 : Assignation de référence

```
1  <?php
2    $rab = 'Pierre';              // Assigne la valeur 'Pierre' à $rab
3    $bar = &$rab;                 // Référence $rab avec $bar.
4    $bar = "Mon nom est $bar";    // Modifie $bar...
5    echo $rab;                    // $rab est aussi modifiée
6    echo $bar;
7    $bar = 'Pierre'
8    echo $bar
9  ?>
```

Ce que le manuel PHP appelle des variables anonymes sont en fait des expressions (combinaisons de valeurs variables ou constantes au moyen d'opérateurs).

Exemple 9 : Assignation de référence & variables anonymes

```
1  <?php
2  $rab = 25;
3  $bar = &$rab; // assignation valide
4  $bar = &(24 * 7); // affectation invalide : expression sans nom
5  function test() {
```

```
6     return 25;
7   }
8   $bar = &test(); // affectation invalide : une fonction n'est pas une variable.
9   ?>
```

Une variable garde sa valeur dans le script où elle est définie tant qu'on ne la change pas. Toutefois compte tenu de la déclaration implicite des variables, quand une variable apparaît dans une fonction, elle supposée spécifique à la fonction & n'est connue que de la fonction. Si on veut employer une variable du script dans la fonction, il faudra le préciser au moyen du mot global.

PAS D'ASSIGNATION DES VARIABLES LOCALES	ASSIGNATION DES VARIABLES LOCALES	UTILISATION DE GLOBAL SANS ASSIGNATION

<?php
$a = 1;
$b = 2;
function somme() {
//global $a, $b;
//$a=5;
//$b=5;
$b = $a + $b;
}
somme();
echo $b, "\n";
?>
	<?php
	$a = 1;
	$b = 2;
	function somme()
	{
	//global $a, $b;
	$a=5;
	$b=5;
	$b = $a + $b;
	}
	somme();
	echo $b, "\n";
	?>
		<?php
		$a = 1;
		$b = 2;
		function somme() {
		global $a, $b;
		//$a=5;
		//$b=5;
		$b = $a + $b;
		}
		somme();
		echo $b, "\n";
		?>

Variables non définies en ligne 8	10	3

La première option affiche une ligne vide, après le message d'avertissement !

Le mot clé static permet de conserver la valeur d'une variable locale. Dans le deuxième exemple ci-dessus, à chaque appel de la fonction *somme()* ***$a*** & ***$b*** sont réinitialisées. Le résultat est toujours

le même. Le mot static permet de conserver la valeur de la variable dans la fonction. Ainsi, au seconde appel de *somme()*, *$b* vaudrait *10*, au lieu de *5*, si *$b* avait été précédé de static.

Il existe des variables prédéfinies. Outre *$this*, il y a des variables, notées en majuscules, dites *superglobales*, car elles sont connues dans tous les scripts & dans toutes les fonctions, sans qu'il soit nécessaire d'employer le mot clé global. La variable *$GLOBAL* contient toutes les autres variables *superglobales*, les trois que nous emploierons le plus sont *$_SERVER* qui contient les variables d'environnement du serveur, *$_POST* & *$_GET* qui permettent d'échanger des données avec les pages web dynamiques par le biais des paramètres de l'URL (la partie de l'URL suivant le).

Exemple 10 : Affichage d'une variable globale

```
1  <?php
2  var_dump($_SERVER["WINDOWMANAGER"]);
3  ?>
```

```
string(17) "/usr/bin/startkde"
```

Il n'est pas nécessaire d'initialiser les variables en PHP, cependant, cela reste une excellente pratique. Les variables non initialisées ont une valeur par défaut selon leur type (**FALSE** pour les booléens, zéro pour les entiers & les réels, chaîne vide pour les chaînes de caractères – comme utilisée avec **echo** – ou un tableau vide pour les tableaux).

Exemple 11 : Valeurs par défaut des variables non initialisées

```
1  <?php
2  // Une variable non initialisée & non référencée (pas de contexte
   d'utilisation); retourne NULL
3  var_dump($nonass_var);
4  // L'utilisation d'un booléen; retourne 'false' (Voir l'opérateur
   ternaire pour comprendre cette syntaxe)
5  echo($nonass_bool ? "true\n" : "false\n");
```

```
6  // Utilisation d'une chaîne de caractères; retourne 'string(3) "abc"'
7  $nonass_str .= 'abc';
8  var_dump($nonass_str);
9  // Utilisation d'un entier; retourne 'int(25)'
10 $nonass_int += 25; // 0 + 25 => 25
11 var_dump($nonass_int);
12 // Utilisation d'un entier/double; retourne 'float(1.25)'
13 $nonass_float += 1.25;
14 var_dump($nonass_float);
15 // Utilisation d'un tableau : retourne array(1) {  [3]=>  string(3)
   "def" }
16 $nonass_arr[3] = "def"; // array() + array(3 => "def") => array(3 =>
   "def")
17 var_dump($nonass_arr);
18 // Utilisation d'un objet; crée un nouvel objet stdClass (voir
   http://www.php.net/manual/fr/reserved.classes.php)
19 // Retourne : object(stdClass)#1 (1) {  ["rab"]=>  string(3) "bar" }
20 $nonass_obj->rab = 'bar';
21 var_dump($nonass_obj);
22 ?>
```

L'utilisation des valeurs par défaut de variables non initialisées est à éviter. Un avertissement *E_NOTICE*, comme celui mentionné dans le tableau précédent, sera émis lorsque vous travaillerez avec des variables non initialisées, cependant, aucune erreur ne sera affichée lorsque vous tenterez d'insérer un élément dans un tableau non initialisé.

Même si la structure de langage isset() peut être utilisée pour détecter si une variable a déjà été initialisée, il est plus rationnel d'initialiser toutes les variables que l'on emploie, selon leur type !

Types

PHP supporte :

- *quatre types simples* :
  - *booléen*, deux valeurs, TRUE & FALSE, ces mots-clés sont insensibles à la casse, une expression logique est une valeur booléenne ;

```
<?php
// assigne d'une valeur booléenne à une variable
$rab = True;
// expression booléenne : == est un opérateur d'égalité qui retourne un booléen, comme tous les opérateurs de comparaison)
if ($action == 'show_version') {
    echo 'C'est vrai !';
}
// Avec une variable booléenne il n'est pas utile d'employer un opérateur d'égalité $var_bool à la même valeur que $var_bool ==True.
if ($show_separators) {
    echo "<hr />\n";
}
?>
```

  - *nombres entiers*

    valeurs comprises entre $-2\ 147\ 483\ 648 = -2^{31}$ & $2\ 147\ 483\ 647 = 2^{31}-1$, le 32e bit servant pour le signe.

    Les entiers peuvent être spécifiés en base décimale (dite aussi base 10), en hexadécimale (base 16) ou octale (base 8). Les entiers peuvent être optionnellement précédés par le signe plus ou moins (- ou +).

    Pour utiliser la notation octale, vous devez préfixer le nombre avec un 0; pour utiliser la notation hexadécimale, vous devez préfixer le nombre avec 0x.

```
1  <?php
2  $a = 178; // nombre entier décimal –base 10–
```

```
3   $b = -178; // nombre entier décimal négatif
4   $c = 0262; // nombre entier octal –base 8– (équivalent à 178 en base 10)
5   $d = 0xB2; // nombre entier hexadécimal –base 16– (équivalent à 178 en base 10)
6   ?>
```

Syntaxe (notation expression rationnelle)

* *decimal* : [1-9][0-9]*| 0
* *hexadecimal* : 0[xX][0-9a-fA-F]+
* *octal* : 0[0-7]+
* *integer* : [+-]?*decimal* | [+-]?*hexadecimal* | [+-]?*octal*

◇ *nombres à virgule flottante*

les valeurs stockées sur 4 (de 1.4E-45 à 3.4028235E38, en négatif & en positif) ou sur 8 octets (de 1.7E-308 à 1.7E+308, en négatif & en positif). Le E ou e signifie *multiplié par 10 à la puissance le nombre qui suit.*

[Rappel : $10^2 = 10 \times 10$, $10^{-2} = \frac{1}{10 \times 10}$ *]*

Syntaxe expression rationnelle

PINT	[0-9]+
NVF	([0-9]*[\.]{*PINT*}) | ({*PINT*}[\.][0-9]*)
NOT_EXPO	[+-]?(({*PINT*} | {*NVF*}) [eE][+-]? {*PINT*})

En clair :

```
<?php
  $a = 1234
  $b = .1234;
  $c = 12.34;
  $d = 1.23e-4;
  $e = -7E5;
?>
```

◇ *chaîne de caractères*

Une chaîne de caractères est une série de caractères, où un caractère est la même chose qu'un octet (De ce fait, *PHP* ne supporte que les jeux de caractères en comportant au plus 256, &, donc, il n'a pas de support natif pour *l'Unicode*.) Sa taille est comprise entre 0 & 2 147 483 647 octets.

Exemple 12 : Notation d'une chaîne de caractères

SCRIPT	RÉSULTAT

`<?php`
`$v=12;`
`// Guillemets simples`
`echo 'Ceci n\'affichera ni nouvelle ligne \n ni contenu de $v';`
`// Guillemets doubles`
`echo "Ceci affichera une nouvelle ligne \n & le contenu $v";`
`// Heredoc : (plusieurs lignes équivalent guillemets doubles)`
`echo <<<FDD`
`\nIl a fait beau le $v, car j'aime la choucroute !\n`
`FDD;`
`// Nowdoc : (plusieurs lignes équivalent guillemets simples)`
`echo <<< 'FDT'`
`Il a fait beau le $v, mais je n'aime plus la choucroute !`
`FDT;`
`?>`
	Ceci n'affichera ni nouvelle ligne \n ni contenu de $v
	
	Ceci affichera une nouvelle ligne
	& le contenu 12
	
	Il a fait beau le 12 !
	& j'aime la choucroute !
	
	Il a fait beau le $v, mais je n'aime plus la choucroute !

- *deux types composés* :

◇ *tableau*, c'est en fait une structure polymorphe, contrairement à ce qui existe dans la plupart des autres langages. En effet, dans ceux-ci, un tableau est une suite limitée de données d'un même type, repérées par un index numérique. En *PHP*, c'est un ensemble illimité de données quelconques repérées par une clé qui peut être numérique ou alphanumérique. De ce fait, il peuvent être employés comme tableaux (suite de données accessibles par leur rang), comme listes (suite dans laquelle on passe d'un élément à l'autre –suivant, précédent), comme table de *hashage* ou tableaux associatifs (suite de données accessibles par une clé), comme piles (dernier entré-premier sorti), comme files (premier entré-premier sorti) [ii], etc.

Un tableau peut être multidimensionnel & contenir d'autres tableaux.

* Il y a deux façons de créer un tableau

Exemple 13 :

```
$taba=array (1=> 'janvier', 'février', ...,'décembre');
$tabb=['déb'=>12, 3=>'dfg', '08'=>'54',];
```

La virgule en fin est facultative. La dernière valeur de $taba sera accessible par $taba[12]. Même dans une déclaration si vous affecter plusieurs valeurs à une même clé, seule la dernière sera retenue.

Les clés sont soumises à quelques contraintes :

* *les chaînes de caractères* contenant un entier valide seront converties en entier ; ainsi la clé "8" sera reconnue comme l'entier 8, mais "08" ne sera pas convertie, car un entier ne peut pas commencer par un "0" ;
* *les nombres à virgule flottante* seront aussi modifiés en entier, ce qui signifie que la partie après la virgule sera tronquée, ainsi la clé "8.7" deviendra 8 ;
* *les booléens* seront modifiés en entier également, i.e. la clé true vaudra 1 & la clé false, 0 ;
* *la valeur Null* deviendra une chaîne vide ("") ;

* *les tableaux & les objets* ne peuvent pas être utilisés comme clés ; si vous le tentez, l'alerte suivante sera émise : Illegal début type.

Exemple 14 :

PHP	RÉSULTAT

`$array = array(`
`  1  => "a",`
`  "1"  => "b",`
`  1.5  => "c",`
`  true => "d",`
`);`
`var_dump($array);`
	`array(1) {`
	`  [1]=>`
	`  string(1) "d"`
	`}`

Il existe de nombreuses fonctions de manipulations de tableaux, nous y reviendrons.

- *objet*

  Les objets PHP sont des objets informatiques traditionnels. On appelle classe la définition d'un type objet & instance, une variable objet. Un objet à des attributs (données caractéristiques, par exemple, longueur, largeur, couleur du fond d'une fenêtre), des méthodes (fonctions spécifiques à l'objet, par exemple ouvrir, fermer, agrandir, rétrécir ou déplacer une fenêtre) & des évènements qu'il subit (arrivée, survol ou départ de la souris, clic ou double clic, frappe d'une touche, etc.). Certains attributs & certaines méthodes peuvent être internes à l'objet, on les dit privés, & d'autres peuvent être connus à l'extérieur de l'objet, on les dit publics.

PHP	RÉSULTAT

`class TypObj{`
`  private $priv = 1;`
`  public $pub = 2;`
`  public function FoncExemp(){`
`  return (object)(array) $this;`
	`// le transtypage array permet de forcer le résultat comme stdClass`

PHP	RÉSULTAT

}
}
$aux = new TypObj();
var_dump($aux->FoncExemp());
 	
var_dump($aux->FoncExemp()->priv);
 	
var_dump($aux->FoncExemp()->pub);
	
	object(stdClass)#2 (2) {
	["priv":"TypObj":private]=>
	int(1)
	["pub"]=>
	int(2)
	}
	
	PHP Notice: Undefined property:
	stdClass::$priv in ...
	NULL
	
	int(2)

Déclaration d'une classe & création d'une instance

PHP propose une classe d'objet par défaut nommée ***stdClass***.

La variable prédéfinie ***this*** référence l'objet courant.

- *deux types spéciaux* :
  - *ressource* : c'est une variable spéciale, contenant une référence vers une ressource externe (un fichier ouvert, une connexion à une base de données, une image, etc.)
  - *NULL* : la valeur spéciale ***NULL*** représente une variable sans valeur; ***NULL*** est la seule valeur possible du type ***NULL***; une variable est considérée comme ***null*** si :
    - elle s'est vue assigner la constante ***NULL*** ;
    - elle n'a pas encore reçu de valeur ;
    - elle a été effacée avec la fonction *unset()*.

PROGRAMMATION ORIENTÉE OBJET (POO)

Comme en C++, les versions actuelles de PHP (à partir de PHP 5) permettent de programmer en orienté objet, en créant des classes contenant des attributs & des méthodes, des instances de classes. L'héritage entre les classes existe aussi. Voici un exemple de création d'une classe :

```
1  <?php
2    class Perso {
3      const PV_initial = 2000;
4      private $PV;
5      public function __construct($type = 'N/A', $PV = 'N/A') { //
   Paramètres optionnels
6        if (!is_numeric($PV) || !($PV > 0 && $PV < 100000000))
7          $this->PV = self::PV_initial;
8        else
9          $this->PV = $PV;
10     }
11     public function getPV() {
12       // Accesseurs
13       return $this->PV;
14     }
15     public function isDead(){
16       return $this->PV == 0;
17     }
18   }
19   // Création d'une classe enfant de Perso
20   class Magicien extends Perso {
21     private $magie;
22   }
23   // Création d'une instance de classe
```

```
24      $perso = new Perso(1000);
25      // Utilisation de l'objet
26      echo 'Votre personnage a ' . $perso->getPV() . ' PV.' .
    PHP_EOL;
27      // Constantes de classes
28      echo 'Le PV par défaut attribué à un nouveau personnage est
    de ' . Perso::PV_initial . '.';
29      // Destruction de l'objet
30      unset($perso);
31  ?>
```

Expressions

Une expression est une combinaison de variables & de constantes au moyen d'opérateur. La forme la plus simple d'expression est une constante ou une variable.

Les opérateurs changent selon le type de la données, ils sont exécutés avec une certaine priorité, ou lorsqu'ils sont de même priorité, selon un certain ordre.

Associativité	Opérateurs	Information additionnelle
non-associatif	clone new	Instance d'objets
gauche	[	array()
droite	**	arithmétique
non-associatif	`	exécution
droite	++ –	incrément/décrément
	(int) (float) (string) (array) (object) (bool)	types
	@	contrôle d'erreur
	~	négation bit à bit

ASSOCIATIVITÉ	OPÉRATEURS	INFORMATION ADDITIONNELLE
non-associatif	instanceof	types
droite	!	négation logique
gauche	* / %	arithmétique
gauche	+ - .	Arithmétique chaîne de caractères
gauche	<< >>	décalage bit à bit
non-associatif	< <= > >=	comparaison
	== != === !== <>	comparaison
gauche	&	ET bit à bit et références
gauche	^	OU exclusif bit à bit
gauche	|	OU bit à bit
gauche	&&	ET logique
gauche	||	OU logique
gauche	? :	ternaire
droite	= += -= *= **= /= .= %= &= |= ^= <<= >>= =>	affectation
gauche	and	ET logique
gauche	xor	OU exclusif logique
gauche	Or	OU logique
droite	(	Rupture de précedence
gauche	,	plusieurs utilisations

Sans compter les expressions rationnelles étendues que PHP gère également.

Instructions

Les instructions sont séparées par des ; (il n'est pas obligatoire après la dernière instruction) & les sauts de ligne ne modifient pas le fonctionnement du programme. Il serait donc possible d'écrire :

```
1 <?php echo 'Hello World';echo 'Comment allez-vous ?';echo 'Il fait beau non ?' ?>
```

Pour des raisons de lisibilité, il est néanmoins recommandé d'écrire une seule instruction par ligne. Il est aussi préférable d'écrire le dernier ;.

Structures de contrôle

Le code PHP est composé d'appel à des fonctions, dans le but d'attribuer des valeurs à des variables, le tout encadré dans des alternatives & des itérations, c'est-à-dire dans des ruptures de l'ordre séquentiel d'exécution.

Exemple 15 :

```
1  <?php
2  // la fonction strtolower renvoie en minuscules la chaîne de
   caractères passée en paramètre
3  $lang = strtolower($_POST['lang']);
4  if ($lang == "fr")
5     echo 'Vous parlez français !';
6  elseif ($lang == "en")
7     echo 'You speak English!';
8  else
9     echo 'Je ne vois pas quelle est votre langue !';
10 ?>
```

Une condition est appliquée quand l'expression entre parenthèses est évaluée à *true* & elle ne l'est pas dans le cas de *false*. Sous forme numérique, *0* représente le *false*, & *1* (& tous les autres nombres) représentent le *true* [iii].

Le code précédent pourrait aussi être écrit de cette manière :

```
1  <?php
2  $lang = strtolower($_POST['lang']);
3  $français = $lang == "fr";
4  $anglais = $lang == "en";
5  if ($français)
6     echo 'Vous parlez français !';
```

```
7   elseif ($anglais)
8     echo 'You speak English!';
9   else
10    echo 'Je ne vois pas quelle est votre langue !'; // Excepté quand on la tire, il est rare de voir une langue !
11  ?>
```

Ici on teste l'égalité entre *$lang* & "fr", mais pas directement dans le **if** : le test retourne un booléen (c'est-à-dire soit *true*, soit *false*) qui est stocké dans la variable *$français*. On entre ensuite cette variable dans le **if** & celui-ci, selon la valeur de la variable, effectuera ou non le traitement.

Aussi, pour les codes précédents, les caractères { & } délimitant les blocs **if**, **elseif** & **else** ont été omis, pour la seule raison que ces blocs ne contenaient qu'une instruction, dans le cas contraire, ils sont obligatoires.

LES STRUCTURES DIVERSES

Ce sont les mots clés **echo**, **print**, **break**, **continue**, **exit**, **goto**, **declare**, **return**, **require**, **require once**, **include** & **include once**.

Les deux premiers **echo** & **print** relèvent plus du mot clé que de la structure puisqu'ils n'entraînent aucune rupture de séquence.

Les quatre suivantes, **break**, **continue**, **exit** & **goto** interrompent l'itération en cours ou la commande **switch**. **exit** arrête carrément le script. Alors que **goto** étiquette permet de sortir d'un itération ou d'une alternative pour reprendre l'exécution à l'instruction suivant étiquette. Cette instruction doit se trouver dans la même contenant le **goto**.

require, **include** & leurs dérivés permettent d'inclure & d'exécuter un script externe.

La structure **declare** introduit une directive de compilation, notion sans utilité pour l'instant ; là aussi, il s'agit plus d'un mot clé que d'une structure !

Les Structures conditionnelles

if [**elif**] [**else**]

```
if (expression)
  commande
[elsif (expression)
  commande]
[else
  commande]
```

L'expression est une expression logique, c'est-à-dire une expression valant true (1) ou false (0), ce peut être une comparaison, une assertion ou une combinaison des deux. Écrire *$var*=true revient à écrire *$var*, puis que la valeur de la comparaison est la même que celle de la variable.

La commande est soit une commande unique soit une commande bloc (ensemble de commandes compris entre des accolades ouvrantes & fermantes –{ & }– ou entre **:** & **endif**).

<?php	<?php
if ($a == 5):	if ($a == 5) {
echo "a égale 5";	echo "a égale 5";
echo "...";	echo "...";
elseif ($a == 6):	} elseif ($a == 6) {
echo "a égale 6";	echo "a égale 6";
echo "!!!";	echo "!!!";

else:	} **else {**
echo "a ne vaut ni 5 ni 6";	echo "a ne vaut ni 5 ni 6";
endif;	}
?>	?>

switch

Cette instruction remplace plusieurs if relatifs à la même variable.

```
switch ($var) {
    case val1:
        [instructions ;]
        [break;]
    case val2:
        [instructions ;]
        [break;]
    …
    case valn:
        [instructions ;]
        [break;]
    default:
        instructions;
}
```

L'instruction break est facultative, en son absence PHP exécute le cas concerné, puis tous les cas suivants ne se terminant pas par **break**. Si un **case** n'est suivi d'aucune instruction, il est traité avec le suivant.

Exemple 16 :

```
1  <?php
2  $bière='Fischer Réserve Ambrée';
```

```php
switch($bière)
{
    case 'Leffe Royal':
    case 'Chimay Tripple':
        echo 'Bières belges';
        break;
    case 'Kœnig Ludwig Dunkle':
        echo 'Bière allemande';
        break;
    case 'Fuller\'s ESB':
    case 'Fischer Réserve Ambrée':
        if ($bière=='Fischer Réserve Ambrée')
          echo 'Bière française';
        else
          echo 'Bière anglaise';
        echo ' ! C\'est une des meilleures !!!';
        break;
    default :
        echo 'Merci de faire un choix...';
    break;
}
echo PHP_EOL;
?>
```

Les Structures itératives

Elles sont quatre : while & do while quand on ignore le nombre d'itérations & for & foreach quand on le connait.

while

L'itération est exécutée tant que l'expression est vraie. Comme l'expression est testée avant l'exécution, l'itération peut n'être jamais

exécutée. Il faut donc s'assurer que l'expression est fausse si l'on souhaite dérouler l'itération. Inversement, il faudra s'assurer dans l'itération qu'elle change de valeur !

```
while (expression)
    commande
```

ou

```
while (expression):
    commandes
endwhile;
```

Le mot commande indique une commande seule ou une commande bloc (ensemble de commandes entre accolades).

Exemple 17 :

```
1  <?php
2    $fini=false;
3    while (! $fini) {
4      echo time(), PHP_EOL;
5     $fini= (time()%10)!=0;
6    }
7  ?>
```

do while

```
do
commande
while (expression) ;
```

L'itération est toujours exécutée au moins une fois, puisque le test de fait après l'exécution.

Exemple 18 : Tir de bataille navale

```
1  <?php
2  $lig=' ABCDEFGHIJ';
3    do {
4      echo 'Où tirez-vous ? ';
```

```
5       $tir = fgets(STDIN,2);
6       $lig_ok=strpos($lig, $tir[0]);
7       $col_ok=($tir[1]>=0) && ($tir[1]<=9);
8     } while (! $lig_ok || ! $col_ok);
9   ?>
```

La fonction fgets récupère une ligne, ici de 2 caractères, du fichier mentionné, ici le clavier (*STDIN*).

for

Cette structure s'emploie quand le nombre d'itérations est connu.

```
for (expr1; expr2; expr3)
  commande
ou
for (expr1; expr2; expr3):
  commandes
endfor;
```

Exemple 19 :

```
1   <?php
2     $pile=0;
3     $face=0;
4     for($col = 'R'; $col != 'AD'; $col++)
5       echo $col,' ';
6      echo PHP_EOL;
7     for ($i=1;$i<=2000;$i++) {
8       if (rand()%2==0)
9         $pile++;
10      else
11        $face++;
12    }
13    echo "pile : ",$pile," *** face : ",$face, PHP_EOL;
14  ?>
```

La fonction rand() génère un nombre aléatoire. Dans le premier for notez l'emploi de d'un caractère comme repère. C'est pratique pour parcourir des cellules de tableur !

foreach

```
foreach (array_expression as $valeur)
  commande
ou
foreach (array_expression as $key => $valeur)
  commande
```

avec bien sûr les formes alternatives en **: endforeach**.

Exemple 20 :

```
1  <?php
2    $tab = array(1, 2, 3, 4);
3    foreach ($tab as &$élém) {
4      $élém = $élém * 2;
5    }
6    unset ($élém) ;
7    var_dump($tab);
8  ?>
```

Résultat

```
array(4) {[0]=>int(2) [1]=>int(4) [2]=>int(6) [3]=>int(8)}
```

Le unset libère la variable temporaire *$élém*, sans lui, elle aurait continué à pointer sur le dernier élément du tableau.

Les Fonctions

Une fonction est un verbe ou un nom verbal du langage. Un verbe exécute une action, un nom verbal aussi, mais en plus il a une valeur.

```
function nom_fonc([$arg[, $arg] ...])
{
```

```
    instructions;
    [return $retval;]
}
```

Ni les fonctions ni leurs arguments ne sont typés, contrairement à d'autres langages.

Vous pouvez rajouter les vôtres, mais auparavant vérifiez qu'elles ne se trouve pas déjà dans PHP, car celui_ci incorpore plus de 2 000 fonctions qui, outre les manipulations des types scalaires se rapportent aux sujets suivants : modification du comportement de PHP, manipulation audio, services d'identification, ligne de commande, l'archivage & la compression, traitement des cartes de crédit, cryptographie, bases de données, dates & heures, systèmes de fichiers, langage humain & encodage de caractères, génération & traitement des images, e-mails, mathématiques, données non-textuelles, contrôle des processus, moteurs de recherche, serveurs, sessions, traitement du texte, variables & types, services Web, Windows uniquement, manipulation XML, autres extensions basiques, autres services. Dans la documentation ces fonctions & leurs arguments apparaissent typés : mais c'est informatif !

Une fonction peut être récursive : dans ce cas, elle s'appelle elle-même.

En voici un exemple classique.

Vous l'ignorez sans doute, mais il y a trois mille de cela Civa construisit dans cette ville, un temple que seuls ses prêtres peuvent voir & pénétrer. Dans ce temple, il installa trois mats d'or massif & sur le premier d'entre-eux, il plaça une pile de 64 disques concentriques & il somma ses prêtres de déplacer cette tour du premier mat au troisième, sans déplacer plus d'un disque à la fois & sans mettre un disque grand sur un disque plus petit. Il leur expliqua qu'ils devaient déplacer, sans relâche, jour & nuit un disque par seconde, que toute erreur serait punie par la mort du fautif & que

la fin du monde & son règne surviendraient quand ils auraient terminé leurs déplacements, à raison d'un disque par seconde. Il s'est dit, au milieu de XIX^{e} siècle que, depuis, 1953 prêtres avaient trouvé la mort, mais le déplacement n'a pas pris de retard.

Un mathématicien français franc-maçon eu vent de l'histoire & il en fit un casse-tête pour ses étudiants. On trouve, même dans le commerce de tels casse-tête, mais avec un nombre plus faible de disques (8 le plus souvent).

Voici un exemple de résolution informatique de ce problème.

Exemple 21 :

```
<?php
function hanoï($matd, $mati, $matf, $nb_disque) {
        if ($nb_disque>=1) {
          $nb_disque--;
          hanoï($matd,$matf,$mati,$nb_disque);
        }
         echo "Déplacer le disque du mat ",$matd," au mat ", $matf,
PHP_EOL;
        if ($nb_disque >= 1) {
          $nb_disque--;
          hanoï($mati, $matd, $matf, $nb_disque);
        }
}
$nb_disque=4;
hanoï(1, 2, 3, $nb_disque);
?>
```

Pour vous rassurer, même si ces fanatiques ne perdent jamais le rythme, il nous reste encore 584 542 043 091 années avant la fin du monde ! D'après nos calculs, il faudrait que le Toshiba Satellite Pro (micro-processeur à 2,41 GHz) sur lequel a été testé ce script, fonc-

tionne, sans interruption, pendant plus de 3 000 ans pour qu'il en arrive à bout.

Autres éléments structurants du PHP

Les objets & leurs classes, les exceptions, les références & les entités seront vues en temps utiles.

Mais il est deux notions nécessaires pour transférer des fichiers ou des données qu'ils nous faut aborder, ce sont les ressources & les variables fichiers. Théoriquement, il n'existe pas de type file, mais il existe des *descripteurs de fichiers* (*files handles*).

Les Ressources

Une ressource est une variable spéciale, contenant une référence vers une ressource externe. Les ressources sont créées et utilisées par des fonctions spéciales.

Voici quelques uns des types de ressources disponibles avec pour celles qui nous intéressent directement (les ressources MySQL), le nom des fonctions permettant de les créer & de les supprimer, plus le nom de quelques fonctions permettant de les manipuler.

Ressource	Définition	
	Construite ou détruite par	Utilisée par
ftp	Flux FTP	
imap	Lien vers un serveur mail (IMAP, POP3)	
ldap link	Lien vers un serveur LDAP	
ldap result	Résultat de recherche LDAP	
mssql link	Lien vers une base de données Microsoft SQL Server	
mssql link persistent	Lien persistant vers une base de données Microsoft SQL Server	

RESSOURCE	DÉFINITION	
	CONSTRUITE OU DÉTRUITE PAR	UTILISÉE PAR
mssql result	Résultat Microsoft SQL Server	
mysql link	Lien vers une base de données MySQL	
	mysql_connect() destructeur mysql_close()	*mysql_create_db(), mysql_nom_bdd(), mysql_db_query(), mysql_drop_db(), mysql_query(), mysql_result(), mysql_select_db(), mysql_tablename(),*
mysql link persistent	Lien persistant vers une base de données MySQL	
	mysql_pconnect() pas de destructeur	*mysql_create_db(), mysql_nom_bdd(), mysql_db_query(), mysql_drop_db(), mysql_query(), mysql_result(), mysql_select_db(), mysql_tablename(),*
mysql result	Résultat MySQL	
	mysql_db_query( )mysql_list_tables (), mysql_query(), destructeur mysql_free_result()	*mysql_nom_bdd(), mysql_fetch_array(), mysql_fetch_assoc(), mysql_fetch_field(), mysql_fetch_longueurs(), mysql_fetch_row(), mysql_field_name(), mysql_field_table(),*
odbc link	Lien vers une base de données ODBC	
Lien persistent odbc	Lien persistant vers une base de données ODBC	
odbc result	Résultat ODBC	
Document pdf	Document PDF	
pdf image	Image dans un document PDF	

Le manuel PHP contient la liste complète des types de ressources & des fonctions permettant de les employer.

Celles qui vont nous intéresser sont, par exemple, les fonctions :

- *mysql_connect()* connexion à une base ;
- *mysql_close()* déconnexion d'une base ;

- *mysql_query()* envoi d'une requête ;
- *mysql_result()* obtention du résultat de la requête ;
- *mysql_fetch_array()* stockage dans un tableau du résultat d'une requête.

Les Descripteurs de fichiers

Ce sont des ressources particulières liées à un protocole de communication. Les quatre protocoles, que nous emploierons, sont :

- file:// accès au système de fichiers local ;
- http(s):// accès aux URL HTTP(S) ;
- ftp(s):// accès aux URL FTP(S) ;
- php:// accès aux divers flux I/O.

L'accès aux fichiers & aux URL se fait par l'ouverture du document avec la fonction fopen(). Dans les trois premiers cas, la chaîne contenant le document à employer commence par l'indication du protocole (file://, http://, ftp://). Dans le dernier, le nom du protocole doit être suivi de ; input, ouput, stdin, stdout, stderr, fd, temp, memory, selon le type de flux désiré (cf. Manuel PHP).

Les fonctions sur les fichiers

À faire

Gestion des erreurs

Dès que l'on tente de lire ou d'écrire des informations sur des périphériques, des problèmes (dispositifs indisponibles, droits sur les fichiers, verrouillage de données, etc.), peuvent apparaître, générant des erreurs pouvant bloquer le script. La gestion des erreurs consiste à permettre un arrêt en douceur du script, plutôt que brutal, en interceptant le message d'erreur.

Il existe en PHP deux méthodes pour y arriver.

La première, simple, repose sur la fonction die(). Elle s'utilise dans les scripts dits procéduraux, c'est-à-dire n'employant pas d'objets.

La seconde, plus complexe emploie des objets & les instructions **throw**, **try**, **catch** ou **finaly**.

À titre d'exemple, nous présenterons un script permettant de se connecter à la base de données *dvd* & de créer une table *emprunteur*, sous la forme procédurale & sous la forme orientée objet. Dans les deux cas, nous supposons que *$host* vaut "localhost", *$login*, "moi_bim", *$password*, "mdp", *$dbname*, "dvd".

OBJET

```php
<?php
  // gest_err_o.php
  try {
    // Connexion
    $c = new PDO("mysql:host=$host;dbname=$dbname",$login,$password);
    echo "Connexion réussie.", PHP_EOL;
    // Création de la table
    $q = "CREATE TABLE emprunteur(Prenom VARCHAR(8) PRIMARY KEY,amande FLOAT)";
    if ($c->exec($q) === FALSE)
      echo "Erreur pendant la création de la table.", PHP_EOL;
    else
      echo "Table créée.", PHP_EOL;
  } catch(PDOException $erreur) {
    echo "Erreur : " . $erreur->getMessage(). PHP_EOL;
  }
?>
```

PROCÉDURES

```php
<?php
  // gest_err_o.php
  // Création de la connexion
  $conn = mysq_connect( $servername, $username, $password) or die("Impossible de se connecter au serveur !" );
  echo "Connexion réussie.", PHP_EOL;
  mysql_select_db ("dvd") or die("Connexion à la base impossible !" );
  echo "Base trouvée !",PHP_EOL;
  // Création de la table
  $q = "CREATE TABLE emprunteur(prenom VARCHAR(8) PRIMARY KEY, amande FLOAT)";
  mysql_query($q) or die("La création de la table a échoué !");
  echo "Table créée.".PHP_EOL;
  close($conn);
?>
```

PDO est la fonction permettant de créer une instance de la classe d'objet permettant de se connecter à un SGBD. Son exécution crée

en cas d'erreur (avec l'instruction **throw**) une instance *PDOException* qui, une fois capturée (instruction **catch**), affichera le message d'erreur en rapport.

Un des intérêts de l'approche objet, dans ce cas, est qu'il suffit de remplacer, en ligne 5, mysql par le nom d'un autre SGBD pour que le script fonctionne alors que, dans l'approche procédurale, il faut modifier les lignes 4, 6 & 10.

La séquence

```
fonction(param) or die (message);
```

est identique à

```
if (fonction(param) === FALSE){ die(message)};
```

puisque le second opérande de l'opérateur **or** n'est évalué que si le premier est faux.

La gestions des erreurs par les objets n'a pas toujours les mêmes conséquences que la gestion procédurale, car la fonction die() comme son nom l'indique arrête le script, alors que la structure **try-catch-finally** poursuit l'exécution au code à l'instruction suivant la dernière accolade. Lorsque **finally** est présente, les instructions qui la suive sont exécutée avant le retour à la séquence appelante.

Dans la jargon PHP die() relève de la gestion d'erreur (*error reporting*) & les séquences **try** de la gestion des exceptions (*exception management*). Une erreur arrête toujours le script, une exception pas toujours.

Génération de code HTML

Un des atouts du *PHP*, sa raison d'être première est la génération de pages web dynamiques. Si l'exécution d'un script ne contenant pas HTML se fait sans problème en ligne de commande, il n'en est pas de même pour ceux en contenant qui génèrent de nombreux messages d'erreur. Pour les exécuter, il est souhaitable d'avoir un serveur web actif, de placer le script *php* à exécuter à la racine du

site web local, de façon à y accéder depuis votre navigateur par l'URL : localhost/nom_du_script.php.

Non seulement les codes HTML & PHP peuvent être mêlés, mais on peut, tout à fait, générer du code HTML dans un script PHP, comme ceci :

```
1  <?php
2    $nom = 'Théophraste';
3    echo '<div>Bonjour ', $nom, ' !</div>';
4  ?>
```

Attention toutefois, la présence de beaucoup de code HTML au sein de code PHP dénote, excepté si vous écrivez une page web dynamique, une mauvaise conception du programme. Dans la partie affichage d'une application web PHP, il faut penser PHP comme un langage de modèle, comme le montre l'exemple de code suivant (utilisant une syntaxe alternative pour la structure **if**/**else**) :

```
1  <?php
2    $nomsAutorises = array('Albert', 'Bertrand');
3    $monNom        = 'Eve';
4  ?>
5  <?php if (in_array($monNom, $nomsAutorises)): ?>
6    <!-- code HTML -->
7    <p>Bonjour <?php echo $monNom ?> !</p>
8  <?php else ?>
9    <!-- code HTML -->
10   <p>Vous n'êtes pas un utilisateur autorisé !</p>
11 <?php endif ?>
```

Une autre approche pour générer du code HTML est de concaténer l'intégralité du code HTML dans une variable & d'afficher la variable en fin de fichier, avec par exemple echo :

```
12 <?php
```

```php
    $code = '';
    // On fait des echos
    $code .= '<p>a' . 'b' . 'c' . "\n";
    $code .= 'd' . 'e' . 'f</p>' . "\n";
    // On affiche le contenu des echos
    echo $code;
?>
```

Remarque : Vous pouvez sauter ce paragraphe, si les précédents vous ont paru trop ardus !

Dans le cas où l'utilisateur aura préféré l'utilisation de la commande echo à la concaténation, il lui sera possible de capturer le flux en utilisant les fonctions ob_start() & ob_get_clean() :

```php
<?php
    // On place le flux dans le buffer de sortie, Le buffer est une variable sans nom qui sert à stocker les informations que l'on veut envoyer au navigateur en une seule fois. Il est traité par des fonctions dont le nom commence par « ob_ » pour open buffer.
    ob_start();
    // On fait des echos dans le buffer
    echo '<p>a', 'b', 'c', "\n";
    echo 'd', 'e', 'f</p>', "\n";
    require_once 'fichier.php'; // Plein de echos dans ce fichier que l'on met dans le buffer
    // On récupère son contenu & on le vide (pour un usage ultérieur)
    $code = ob_get_clean();
    // On affiche le contenu du buffer dans le navigateur.
    echo $code;
?>
```

Remarque : vous pouvez sauter ce paragraphe, si les précédents vous ont semblé trop ardus !

PHP, tout comme JavaScript, permet aussi de construire un modèle objet de document (DOM), ce qui permet de créer ou modifier un document [X]HTML sans écrire de HTML, comme le montre l'exemple suivant où *$doctype* est un objet de la classe DOM, *$domt* l'objet, contenu dans *$doctype* & contenant la page HTML & *$html*, l'objet contenu dans la page HTML qui contient le code HTML :

```
1  <?php
2      $doctype =
   DOMImplementation::createDocumentType('html',"-//W3C
   //DTDHTML4.01//EN","http://www.w3.org/TR/html4/strict.dtd");
3      $domt = DOMImplementation::createDocument(null, "html",
   $doctype);
4      $html = $domt->documentElement;
5      $html->head = $domt->createElement("head");
6      $html->appendChild($html->head);
7      $html->head->title = $domt->createElement("title");
8      $html->head->title->nodevaleur = "Exemple de HTML";
9      $html->head->appendChild($html->head->title);
10     $html->head->charset = $domt->createElement("meta");
11     $html->head->charset->setAttribute("http-equiv", "Content-
   Type");
12     $html->head->charset->setAttribute("content", "text/html;
   charset=utf-8");
13     $html->head->appendChild($html->head->charset);
14     $html->body = $domt->createElement("body");
15     $html->appendChild($html->body);
16     $html->body->p = $domt->createElement("p");
17     $html->body->p->nodevaleur = "Ceci est un paragraphe.";
18     $html->body->appendChild($html->body->p);
```

```
19      $html->body->p->br = $domt->createElement("br");
20      $html->body->p->appendChild($html->body->p->br);
21      $html->body->p->a = $domt->createElement("a");
22      $html->body->p->a->nodevaleur = "Ceci est un lien.";
23      $html->body->p->a->setAttribute("href", "cible.html");
24      $html->body->p->appendChild($html->body->p->a);
25      print($domt->saveHTML());
26  ?>
```

Qui crée le code HTML suivant :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Exemple de HTML</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body><p>Ceci est un paragraphe.<br><a href="cible.html">Ceci est
un lien.</a></p></body>
</html>
```

Tel qu'il est présenté cet exemple est idiot, puisqu'il est plus rapide de saisir directement le HTML. Si, en revanche vous stockez ces informations dans une base de données & que vous les rappelez ultérieurement, vous aurez créé une page web réellement dynamique.

Configuration

La page http://php.net/manual/fr/configuration.file.php & les pages auxquelles elle est liée présentent exhaustivement le fichier de configuration php.ini. Sur notre système OpenSuse la configuration est contenue dans le dossier /etc/php5. Le fichier de configuration prin-

cipal se trouve des /etc/php5/Apache2. Les fichiers de configuration des extensions dans le dossier /etc/php5/conf.d & chaque fichier comporte deux lignes.

Exemple 22 : mysql.ini

```
; comment out next line to disable mysqli extension in php
extension=mysqli.so
```

Le fichier php.ini commencepar une section [PHP] contenant son mode d'emploi, sous forme de lignes commentées,

Exemple 23 :

```
[PHP]
;;;;;;;;;;;;;;;;;;;
; About php.ini   ;
;;;;;;;;;;;;;;;;;;;
; PHP's initialization file, generally called php.ini, is responsible for
; configuring many of the aspects of PHP's behavior.
; PHP attempts to find and load this configuration from a number of
locations.
; The following is a summary of its search order:
; 1. SAPI module specific location.
; 2. The PHPRC environment variable. (As of PHP 5.2.0)
; 3. A number of predefined registry keys on Windows (As of PHP
5.2.0)
; 4. Current working directory (except CLI)
; 5. The web server's directory (for SAPI modules), or directory of
PHP
[...]
```

puis les directives générales

Exemple 24 :

```
; Enable the PHP scripting language engine under Apache.
; http://php.net/engine
```

```
engine = On
[...]
```

& s'achève par des sections ini spécifiques aux extensions standards.

Exemple 25 :

```
[Pdo_mysql]
; If mysqlnd is used: Number of cache slots for the internal result set
cache
; http://php.net/pdo_mysql.cache_size
pdo_mysql.cache_size = 2000

; Default socket name for local MySQL connects.  If empty, uses the
built-in MySQL defaults.
; http://php.net/pdo_mysql.default-socket
pdo_mysql.default_socket=
```

Les URL indiques la page du manuel en ligne décrivant la directive concernée.

Informations diverses

Mascotte

PHP est souvent accompagné de l'éléPHPant, dessiné par El Roubio. Il s'est inspiré de la ressemblance des lettres PHP avec un éléphant, d'où le nom. Toutes ses œuvres sont distribuées sous licence GNU GPL. Il existe aussi des origamis & des peluches ÉléPHPant.

Quelques exemples d'application

* Wiki (MediaWiki, WikiNi, DokuWiki...)
* Forums (phpBB, IPB, punBB...)
* Systèmes de gestion de blog (Dotclear, WordPress...)

* Systèmes de gestion de contenu (appelés aussi CMS) –Spip, Drupal, Xoops...)
* Administration de bases de données (phpMyAdmin, phpPgAdmin, Eskuel...)
* Frameworks (Zend Framework, CakePHP, Symfony, etc.)

Accélération

PHP est à la base un langage interprété, ce qui est au détriment de la vitesse d'exécution du code. Sa forte popularité associée à son utilisation sur des sites web à très fort trafic (Yahoo, Facebook) ont amené un certain nombre de personnes à chercher à améliorer ses performances pour pouvoir servir un plus grand nombre d'utilisateurs de ces sites web sans nécessiter l'achat de nouveaux serveurs.

La réécriture du cœur de *PHP* ayant abouti au *Zend Engine* pour *PHP 4* puis le *Zend Engine 2* pour *PHP 5* sont des optimisations. Le *Zend Engine* compile en interne le code PHP en bytecode exécuté par une machine virtuelle. Les projets open source APC & eAccelerator fonctionnent en tant que cache pour accélérer encore un peu la génération des pages web.

Il existe également des projets pour compiler du code PHP :

- Roadsend & phc compilent du PHP en C,
- Quercus compile du PHP en bytecode Java exécutable sur une machine virtuelle Java,
- Phalanger compile du PHP en Common Intermediate Language exécutable sur le Common Language Runtime du framework *.NET*.
- HipHop For PHP transforme du PHP en C++ qui est ensuite compilé en code natif. Ce projet open source a été démarré par Facebook.

À notre niveau, ce problème de rapidité relève de l'hébergeur.

Exercices

Ex. 01

Ex. 02

Ex. 03

Ex. 04

Ex. 05

Ex. 01

Ex. 06

Ex. 07

SQL

Nous allons employer le même schéma d'exposition que pour le PHP. Après une brève introduction, définissant les concepts de bases, nous commenceront par les valeurs littérales, puis par les identifiants, en continuant avec les expressions & en finissant par les instructions.

SQL (sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

Outre le langage de manipulation des données, la partie langage de définition des données permet de créer et de modifier l'organisation des données dans la base de données, la partie langage de contrôle de transaction permet de commencer et de terminer des transactions, et la partie langage de contrôle des données permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes.

Créé en 1974, normalisé depuis 1986, le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégé SGBDR) du marché.

Nous présenterons le SQL-99 utilisé par MariaDB

LES BASES DU LANGAGE

Après avoir défini le vocabulaire, nous aborderons la syntaxe du SQL, puis ses fonctions, son interface avec PHP & nous finirons avec l'emploi du client PHPMyAdmin.

VOCABULAIRE

L'annexe SQL-1 contient les définitions commentées de tous les mots utiles, voici une présentation simplifiée des plus employés.

Une *relation* ou *table* est l'ensemble des caractéristiques d'un objet. Elle peut être permanente ou provisoire, nommée ou non, stockée en mémoire volatile ou permanente.

Objet est synonyme de définition d'une *table*.

Caractéristique ou *attribut* est synonyme de *colonne,* de *champ* ou de *rubrique*.

Une *ligne* est la réalisation d'un objet, c'est-à-dire les valeurs particulières liées des différentes caractéristiques. On dit aussi *fiche* ou *enregistrement.* Quelquefois on parle de *tuple*.

Il y a une confusion fréquente de sens entre la notion de relation ci-dessus & les liens entre les tables.

Un *index* est une relation utilisée & entretenue par le SGBD pour lui permettre de retrouver rapidement les données. Son emploi simplifie & accélère les opérations de recherche, de tri, de liens ou d'agrégation de données effectuées par le SGBD.

Une *clé* désigne la colonne de la table index (*relation index*) ayant des valeurs communes avec la table de données, mais aussi la définition de l'index ou l'identifiant d'une ligne de cet index.

Une *clé primaire* est associée à une ligne unique de la table de données & elle sert, par défaut, pour accéder aux données de la relation. En d'autres termes, au lieu d'être présentées dans l'ordre chro-

nologique de saisie, qui, souvent, ne présente aucun intérêt, elles seront triées dans un ordre facilitant leur lecture.

Les Éléments du langage

Ce langage manipule six sortes de constituant des phrases.

Les clauses sont des composants des instructions & des requêtes. Elles sont souvent optionnelles. Ordre est un synonyme à la fois d'instruction & de requête.

Les expressions, produisnet soitent des valeurs scalaires (nombres, chaînes de caractères) soit des tables.

Les prédicats spécifient des conditions évaluées soit dans la logique à trois valeurs du SQL (TRUE/FALSE/UNKNOWN) soit en logique booléenne (TRUE/FALSE). Ils servent à limiter les effets des instructions & des requêtes ou à modifier le déroulement des programmes.

Les requêtes, qui extraient des données en fonctions de critères spécifiques sont un élément important du langage.

Les instructions ont un effet permanet sur l'organisation d'une base ou sur ses données, elles peuvent contrôler les transactions, le déroulement des procédures, les sessions & les diagnostics. Elles se terminent par un point-virgule ;. Il arrive que ce dernier ne soit pas obligatoire sur certaines plateformes.

Les espaces insignifiants sont généralement ignorés, ils serevent à faciliter la lecture du code.

Les relations

Donc, une relation ou table comporte des colonnes dont le nom & le type caractérisent le contenu qui sera inséré dans la table.

Ainsi dans notre base *DVD* nous caractérisons, dans une table *Personnes*, les acteurs, les réalisateurs, les compositeurs & les auteurs par leur nom, par leur prénom, par leur fonction, par leur

date de naissance, leur sexe & leur nationalité. Afin de faciliter la recherche nous rajouterons une colonne code qui comportera la première lettre de leur fonction (A pour acteur, R pour réalisateur, C pour compositeurs, a pour auteurs), la première & la dernière lettre de leur nom & la première lettre de leur prénom, suivies de deux chiffres pour différencier ceux ayant le même début.

Exemple 26 : réalisateur Fritz Lang

Le code sera RLGF01.

Ce code sera employé comme clé primaire.

Nom	Prénom	Fonction	Codepers	dat_naiss	naionalité	genre
Faire glisser pour réordonner						
Lang	Fritz	R	RLGF01	2013-11-03	All	Homme

Nous ne garantissons pas la date de naissance.

L'ALGÈBRE RELATIONNELLE.

Ce un grand mot nous permet d'introduire trois notions :

- *projection*, sélection d'un ou plusieurs attributs d'une relation (on ignore les autres) ; par exemple, n'afficher que les colonnes *nom* & *prénom* de la table *Personnes* ;
- *jointure,* création d'une nouvelle relation à partir de deux ou plusieurs autres en prenant comme pivot un ou plusieurs attributs ; par exemple, on concatène la table *Coffrets* avec *Personnes* pour celles qui sont des réalisateurs (c'est typiquement du recoupement de fichiers) ;
- *sélection,* extraction de tous ou parties des lignes en fonction de critères de sélection portant sur les valeurs des attributs ; par exemple, n'afficher que les lignes de la table *Personnes* qui vérifient la condition suivante : *nom* ne commence pas par la lettre C.

Ces trois opérations sont facilement réalisables avec la commande SQL **SELECT** & ses clauses (FROM, WHERE, etc.)

- *Projection d'une table*

Titre	Achat	Genre	Pays	ccoffret	Nb_dvd	personne	cperson
2001 : l'odyssée de l'espace	O	F	A	C051	1	Kubrick S.	
3 comédies de Molière (Cocu imaginaire, école des maris, précieuses ridicules)	O	H	F	C052	1	Schiaretti C.	
3 hommes & 1 couffin	O	Hu	F	C053	1	Serreau C.	

```
SELECT `Titre`, `Genre`, `Pays` FROM `Coffrets`;
```

Titre	Genre	Pays
2001 : l'odyssée de l'espace	F	A
3 comédies de Molière (Cocu imaginaire, école des maris, précieuses ridicules)	H	F
3 hommes & 1 couffin	Hu	F

* *Jointure de deux tables*

Avec la table *Coffrets*

Titre	Achat	Genre	Pays	ccoffret	Nb_dvd	personne	cperson
2001 : l'odyssée de l'espace	O	F	A	C051	1	Kubrick S.	
3 comédies de Molière (Cocu imaginaire, école des maris, précieuses ridicules)	O	H	F	C052	1	Schiaretti C.	
3 hommes & 1 couffin	O	Hu	F	C053	1	Serreau C.	

& avec la table *Catég*

codecat	libellé
F	Science-fiction
Fa	Fantastique
Gu	Guerre
Hi	Historique
Hu	Humour

avec l'instruction

```
SELECT `Coffrets`.`Titre` , `Catég`.`libellé` FROM Coffrets, Catég
WHERE ( `Coffrets`.`Genre` = `Catég`.`codecat` )
```

on obtient la table

Titre	libellé
2001 : l'odyssée de l'espace	Science-fiction
3 hommes & 1 couffin	Humour
300	Historique

* *Sélection dans une table*

Titre	Achat	Genre	Pays	ccoffret	Nb_dvd	personne	cperson
2001 : l'odyssée de l'espace	O	F	A	C051	1	Kubrick S.	
3 comédies de Molière (Cocu imaginaire, école des maris, précieuses ridicules)	O	H	F	C052	1	Schiaretti C.	
3 hommes & 1 couffin	O	Hu	F	C053	1	Serreau C.	

avec l'instruction

```
SELECT * FROM `Coffrets` WHERE `Genre` = "H"
```

Titre	Achat	Genre	Pays	ccoffret	Nb_dvd	personne	cpe
3 comédies de Molière (Cocu imaginaire, école des maris, précieuses ridicules)	O	H	F	C052	1	Schiaretti C.	

LA SYNTAXE DU SQL

Les noms des bases, relations, attributs, index & alias sont constitués de caractères alphanumériques & des caractères _ & $.

Un nom comporte au maximum 64 caractères.

Comme les bases de données & les relations sont codées directement dans le système de fichiers, la sensibilité à la casse de MySQL dépend de celle du système d'exploitation sur lequel il repose. Avec Windows, la casse n'a pas d'importance ; alors qu'avec Unix, elle en a !

Il est de règle d'entourer ces identificateurs avec le caractère `.

Le point, ., est un caractère réservé utilisé comme séparateur entre le nom d'une base & celui d'une relation, entre le nom d'une relation & celui d'un attribut.

Il est d'usage de protéger les noms, mais ce n'est obligatoire que si le nom correspond à un mot déjà défini dans le SGBD. Ainsi on pourra écrire *base1.table25.col5*, mais on devra écrire *`base`.`table`.`col`*.

MOTS RÉSERVÉS

Comme tous les langages le *SQL* incorpore des mots réservés, mais contrairement au PHP ceux-ci sont très nombreux (L'annexe SQL-2 présente tous les mots réservés du langage SQL-99) ; en voici quelques-uns : ABSOLUTE, ADD, AGGREGATE, ALIAS, ALL, ALTER, AND, ANY, AS, ASC, *BINARY*, *BIT*, *BLOB*, *BOOLEAN*, BOTH, BY, CALL, CAST, *CHAR*, COLUMN, CONSTRAINT, CREATE, *DATE*, *DEC*, *DECIMAL*, DECLARE, DEFAULT, DELETE, DISTINCT, *DOUBLE*, DROP, FALSE,

FLOAT, FOREIGN, FROM, FULL, GROUP, HAVING, IN, INNER, INSERT, *INT*, *INTEGER*, INTO, IS, JOIN, KEY, LEFT, LIKE, LIMIT, NOT, NULL, NUMERIC, OF, ON, OR, ORDER, OUTER, PRIMARY, *REAL*, RIGHT, ROW, SELECT, *SET*, *SMALLINT*, TABLE, TO, TRUE, UNION, UNIQUE, UPDATE, USING, valeur, valeurS, *VARCHAR*, WHERE.

Certains mots sont des instructions (SELECT, INSERT, etc.), d'autres des clauses (WHERE, ORDER BY, etc.), , d'autres des noms de type de données (INTEGER, BLOB, etc.)

Un ordre est une instruction, ou commande, complète (intégrant une ou plusieurs clauses) à exécuter.

OPÉRATEURS

Outre les opérateurs arithmétiques classiques, le SQL possède des opérateurs plus subtils permettant d'élaborer des requêtes complexes.

De la plus basse à la plus haute priorité (*précédence* en jargon) ce sont les suivants :

Niv.	symbolee	Signification
ASSIGNATION		
0	:=	Assignation d'une valeur scalaire mysql> SELECT @motdepass:="a2ed(-rft!74,u"; -> 1
LOGIQUE BOOLÉENNE (T,F) OU SQL (T,F,N)		
1	||, OR	OU logique
	XOR	OU exclusif logique
2	&&, AND	ET logique
COMPARAISON SQL (FONCTIONS DE CONTRÔLE)		
3	BETWEEN	Compris dans mysql> SELECT Cout BETWEEN 100.00 AND 500.00 ;

Niv.	symbolee	Signification
		-> 1
	CASE WHEN THEN ELSE	Si, Quand, Alors,Sinon ou Selon Une des originalités du SQL CASE valeur WHEN [compare-valeur] THEN résultat [WHEN [compare-valeur] THEN résultat ...] [ELSE résultat] END, CASE WHEN [condition] THEN résultat [WHEN [condition] THEN résultat ...] [ELSE résultat] END mysql> SELECT CASE 1 WHEN 1 THEN "un" WHEN 2 THEN "deux" ELSE "plus" END; -> "un"
		COMPARAISON CLASSIQUE & SQL
4	=	Égal à mysql> SELECT Acteur = 'Lémondevin' ; -> 1
	<=>	Comparaison presque identique au précédent, la différence est l'opérateur retourne vrai quand les deux opérandes sont NULL & faux si un seul des deux l'est.
	>=	Supérieur ou égal à mysql> SELECT CA >= 30000.00; -> 1
	>	Supérieur à mysql> SELECT DateNaissance > '1997-03-31'; -> 1
	<=	Inférieur ou égal à
	<	Inférieur à
	<>, !=, DISTINCT	Différent de (!= n'est pas accepté par tous les SGBD). Distinct s'applique aux valeurs non scalaires mysql> SELECT CodePostal <> '38000'; -> 1
	IS	Compare

Niv.	symbolee	Signification
		mysql> SELECT Addressel IS NULL; -> 1
	LIKE	Contenant mysql> SELECT First_Name LIKE 'Will%'; -> 1
	REGEXP	mysql> SELECT 'a' REGEXP '^[a-d]'; -> 1
	IN	Dans une liste mysql> SELECT CodeDept IN (1, 7, 26, 38, 69); -> 1
BIT À BIT 29=11101, 15=01111, 13=01101, 18=10010		
5	|,^	OU bit à bit & OU exclusif bit à bit mysql> SELECT 29 | 15; -> 31 mysql> SELECT 29 ^ 15; -> 18
6	&	ET bit à bit mysql> SELECT 29 & 15; -> 13
7	<<	Décalage à gauche d'un bit ou plusieurs bits mysql> SELECT 1 << 2; -> 4
	>>	Décalage à droite d'un bit ou plusieurs bits mysql> SELECT 4 >> 2; -> 1
ARITHMÉTIQUE		
8	-, +	soustraction & addition
9	*	multiplication
	/, DIV	DIV division entière, / division décimale

Niv.	symbolee	Signification
	%, MOD	reste de la division entière
10	^	exponentiation
	UNAIRES	
11	-	moins unaire
	~	inversion bit à bit mysql> SELECT ~ 29; -> 18446744073709551586 Les entiers sont codés sur 64 bits (sur 5 bits 00010=2)
12	NOT, !	Négation (non logique) mysql> SELECT Addressel IS NOT NULL -> 0
13	BINARY COLLATE	Ce ne sont pas à proprement parler des opérateurs, mais BINARY convertit un nombre en chaîne binaire & COLLATE modifie le jeu de caractère.

Les parenthèses modifient la précédence.

TYPAGE DES DONNÉES

Les propriétés des attributs (typage des données) peuvent être :

- nombre entier signé (température) ou non (quantité commandée, âge),
- nombre à virgule (prix, taille),
- chaîne de caractères (nom, adresse, article de presse),
- date & heure (date de naissance, heure de parution),
- énumération (une couleur parmi une liste prédéfinie),
- ensemble (une ou des monnaies parmi une liste prédéfinie).

Il vous faudra choisir le typage adapté à vos besoins, combinant minimisation de la taille, maximisation de la rapidité de traitement & du confort de programmation.

Ces types requièrent une plus ou moins grande quantité de données à stocker, mais moins de 256 octets, dans tous les cas. Il existe d'autres types permettant de stocker les informations d'une taille supérieure.

Les Types d'entier

Type	Min	Max	Taille
TINYINT	-128	127	1
TINYINT UNSIGNED	0	255	1
SMALLINT INT2	-32768	32767	2
SMALLINT UNSIGNED INT2 UNSIGNED	0	65535	2
MEDIUMINT	-8388608	8388607	3
MEDIUMINT UNSIGNED	0	16777215	3
INT INT4 INTEGER	-2147483648	2147483647	4
INT UNSIGNED INT4 UNSIGNED INTEGER UNSIGNED	0	4294967295	4
BIGINT INT8	-9223372036854775808	9223372036854775807	8
BIGINT UNSIGNED INT8 UNSIGNED	0	18446744073709551615	8

Trois remarques :

- INT est synonyme de INTEGER ;
- UNSIGNED permet d'avoir un type non signé ;
- ZEROFILL autorise l'emploi zéros non significatifs, comme dans le code postal (01000).

Ce qu'il faut retenir : Avant de choisir un type entier, il faut se poser les questions suivantes :

- ai-je besoin de stocker des nombres négatifs ?
- quel est le plus grand nombre que je devrai stocker ?

LES TYPES DITS DÉCIMAUX

Les nombres à virgule, ne sont pas les nombres décimaux au sens mathématique du terme, mais ils en ont l'aspect. Ils sont, encore moins, les nombres réels au sens mathématique du terme, bien qu'on les désigne parfois comme tels. On les appelle, également, FLOAT, REAL, DOUBLE ou DECIMAL. L'expression de nombre à virgule flottante décrit le fait que le nombre de décimales est variable (& non que la virgule reste en surface, alors que les chiffres coulent, quand on plonge un nombre dans l'eau). De toute façon, il est impossible de les représenter avec une précision absolue.

Outre 0, ces nombres varient dans les valeurs précisées dans le tableau suivant.

NOM	VALEURS NÉGATIVES	VALEURS POSITIVES
FLOAT[(M,D)]	*-3.402823466E+38* à *-1.175494351E-38*	*1.175494351E-38* à *3.402823466E+38*
DOUBLE[(M,D)] DOUBLE PRECISION[(M,D)] REAL[(M,D)]	*-1.7976931348623157E+308* à *-2.2250738585072014E-308*	*2.2250738585072014E-308* à *1.7976931348623157E+308*
DECIMAL[(M[,D])] DEC[(M[,D])] NUMERIC[(M[,D])] FIXED[(M[,D])]	Un nombre à virgule flottante noté sous forme de caractères. Il se comporte comme une colonne de type CHAR. La virgule décimale & le signe moins '–' des nombres négatifs ne sont pas comptés dans M (mais de l'espace leur est réservé). Si D vaut 0, les valeurs n'auront pas de virgule décimale ou de partie décimale. L'intervalle de validité du type DECIMAL est le même que DOUBLE, mais il peut être restreint par le choix des valeurs de M & D. Si UNSIGNED est spéci–	

	fié, les valeurs négatives sont interdites. Si **D** est omis, la valeur par défaut est 0. Si **M** est omis, la valeur par défaut est 10. Leur raison d'être s'avère l'absence d'erreur d'arrondis dans les calculs, leur défaut principal, la lenteur des calculs !

Comme pour les types entiers, on peut rajouter les mots UNSIGNED pour interdire les valeurs négatives & ZEROFILL pour autoriser l'emploi zéros non significatifs.

Les données caractères

Elles sont de cinq sortes : chaînes, binaires, données de grande taille, énumérations & ensembles.

Les chaînes de caractères

Les types CHAR & VARCHAR sont similaires, mais ils diffèrent dans la manière dont ils sont stockés & récupérés. Leur déclaration se fait avec la syntaxe CHAR(M) ou VARCHAR(M) où M est le nombre de caractère de la chaîne.

La longueur d'une colonne CHAR est fixée à la longueur que vous avez défini lors de la création de la table. La longueur peut être n'importe quelle valeur entre 1 & 255. Quand une valeur CHAR est enregistrée, elle est complété à droite avec des espaces jusqu'à atteindre la valeur fixée. Quand une valeur de CHAR est lue, les espaces en trop sont retirés.

Les valeurs contenues dans les colonnes de type VARCHAR sont de tailles variables. Vous pouvez déclarer un champ VARCHAR pour que sa taille soit comprise entre 1 & 255, exactement comme pour les rubriques CHAR. Par contre, contrairement à CHAR, les valeurs de VARCHAR sont stockées en utilisant autant de caractères que nécessaire, plus un octet pour mémoriser la longueur. Les valeurs ne sont pas complétées. Au contraire, les espaces finaux sont supprimés

avant stockage (ce qui ne fait pas partie des spécifications ANSI SQL).

Si vous assignez une chaîne de caractères qui dépasse la capacité de la colonne CHAR ou VARCHAR, celle ci est tronquée jusqu'à la taille maximale du champ.

Le tableau suivant illustre les différences entre les deux types de colonnes en montrant les différences entre l'enregistrement dans une colonne CHAR(4) ou VARCHAR(4) :

VALEUR SAISIE	CHAR(4)	OCTETS	VARCHAR(4)	OCTETS
''	' '	4	''	1
'ab'	'ab '	4	'ab'	3
'abcd'	'abcd'	4	'abcd'	5
'abcdefgh'	'abcd'	4	'abcd'	5

Les valeurs lues dans les colonnes de type CHAR(4) & VARCHAR(4) seront les mêmes dans tous les cas, car les espaces finaux sont retirés des valeurs issues de colonnes de type CHAR lors de la lecture.

LES BINAIRES

Les types BINARY & VARBINARY sont similaires à CHAR & VARCHAR, hormis le fait qu'ils contiennent des chaînes binaires, plutôt que des chaînes de texte. C'est à dire, qu'ils contiennent des chaînes d'octets, plutôt que des chaînes de caractères. Cela signifie qu'ils n'ont pas de jeu de caractères associé, & que les tris & les comparaisons sont basés sur la valeur numérique de l'octet.

La taille maximale pour les types BINARY & VARBINARY, est la même que celles de CHAR & VARCHAR, hormis le fait que la taille de BINARY & VARBINARY est une taille en octets, & non pas en caractères.

LES DONNÉES DE GRANDE TAILLE

Il s'agit soit de textes long (TEXT) soit d'informations non textuelles telles les images, les vidéos, les sons & les exécutables, on parle alors de *BLOB* (*Binary Large Object*).

Une valeur de type BLOB est un objet binaire de grande taille, qui peut contenir une quantité variable de données. Les quatre types BLOB (TINYBLOB, BLOB, MEDIUMBLOB & LONGBLOB) ne diffèrent que par la taille maximale de données qu'ils peuvent stocker. De même, les quatre types TEXT (TINYTEXT [iv], TEXT, MEDIUMTEXT & LONGTEXT) correspondent aux types BLOB équivalents, & ont les mêmes contraintes de stockage. Les seules différences entre les colonnes de type BLOB & celles de type TEXT se situent aux niveau des tris & comparaisons : les tris, faits sur les BLOB, contrairement à ceux faits sur les TEXT, tiennent compte de la casse. En d'autres termes, une valeur TEXT est une valeur BLOB insensible à la casse.

Si vous assignez une valeur trop grande à une colonne de type BLOB ou TEXT, la valeur sera tronquée à la taille maximale possible.

Dans la majorité des cas, vous pouvez considérer une colonne de type TEXT comme une colonne de type VARCHAR, aussi grande que vous le souhaitez. De même, vous pouvez considérer une colonne de type BLOB comme une colonne de type VARBINARY. La seule différence est que les TEXT sont insensibles à la casse. En clair, dans un objet TEXT, indépendamment de l'*interclassement* (*collation*), A & a sont classés ensembles alors que, dans un BLOB, le a est classé après le Z.

Pour les index des rubriques BLOB & TEXT, vous devez spécifier une taille d'index. Pour les colonnes de type CHAR & VARCHAR, la taille du préfixe est optionnelle.

Il n'y a pas de suppression des espaces finaux lors du stockage de valeur dans des champs de type BLOB & TEXT, ce qui est le cas dans pour les colonnes de type VARCHAR.

Les colonnes BLOB ou TEXT ne peuvent pas avoir de valeur par défaut (DEFAULT).

LES ÉNUMÉRATIONS

Une énumération ENUM est une chaîne dont la valeur est choisie parmi une liste de valeurs autorisées lors de la création de la table.

Cette chaîne peut aussi être la chaîne vide ("") ou NULL dans certaines circonstances.

- Si vous insérez une valeur illégale dans une énumération ENUM (c'est à dire, une chaîne qui n'est pas dans la liste de valeurs autorisées), la chaîne vide est insérée pour représenter une erreur. Cette chaîne peut être distinguée d'une chaîne vide 'normale' par le fait que cette chaîne à la valeur numérique 0. Nous reviendrons sur ce point plus tard.
- Si une colonne d'énumération est déclarée NULL, NULL devient aussi une valeur autorisée, & la valeur par défaut est alors NULL. Si une colonne d'énumération est déclarée NOT NULL, la valeur par défaut est le premier élément de la liste des valeurs autorisées.

Chaque élément de l'énumération dispose d'un index.

- Les valeurs de la liste des valeurs autorisées sont indexées à partir de *1*.
- L'index de la chaîne vide (cas d'erreur) est *0*. Cela signifie que vous pouvez utiliser la sélection suivante pour repérer les valeurs d'énumération invalides :

```
mysql> SELECT * FROM nom_de_table WHERE enum_col=0;
```

L'index de la valeur NULL est NULL.

Par exemple, une colonne créée comme ENUM("un", "deux", "trois") peut prendre n'importe quelle valeur ci-dessous. L'index de chaque valeur est aussi présenté :

VALEUR	INDEX
NULL	NULL
""	0

VALEUR	INDEX
"un"	*1*
"deux"	*2*
"trois"	*3*

Une énumération peut avoir un maximum de 65 535 éléments.

LES ENSEMBLES

Un SET est une chaîne qui peut avoir zéro ou plusieurs valeurs, chacune doit être choisie dans une liste de valeurs définies lors de la création de la table. Les valeurs des colonnes SET composées de plusieurs membres sont définies en séparant celles-ci avec des virgules (‘,’). Ce qui fait que la valeur d’un membre de SET ne peut contenir lui même de virgule.

Par exemple, une colonne définie en tant que SET("un", "deux") **NOT NULL** peut avoir l’une de ces valeurs :

- ""
- "un"
- "deux"
- "un,deux"

Un SET peut avoir au plus 64 membres.

À partir de la version 3.23.51, les espaces en trop sont automatiquement effacés des membres de SET lorsque la table est créée.

MySQL enregistre les valeurs de SET numériquement. Le bit de poids faible de la valeur correspond alors au premier élément de la liste. Si vous utilisez une valeur SET dans un contexte numérique, les bits des éléments dans cet ensemble seront mis à un, & les autres à zéro. Par exemple, vous pouvez obtenir un entier à partir d’un ensemble comme ceci :

```
mysql> SELECT col_set+0 FROM nom_de_table;
```

Si un nombre est enregistré dans une colonne SET, les bits à un de ce nombre représenteront les éléments placés dans cet ensemble. Si une colonne est spécifiée en tant que SET("a","b","c","d"), les membres ont alors les valeurs suivantes :

SET membre	Valeur décimale	Valeur binaire
a	1	1
b	2	10
c	4	100
d	8	1000

Si vous assignez *9* à cette colonne, soit *1001* en binaire, les valeurs du premier & du quatrième membres "a" & "d" sont sélectionnés & la valeur résultante est "a,d".

Pour les valeurs se composant de plus d'un membre du SET, l'ordre des membres n'a pas d'importance lors des insertions. Le nombre d'occurrence d'un élément n'importe pas non plus. Lorsque la valeur sera lue ultérieurement, chaque élément n'apparaîtra qu'une seule fois, & dans l'ordre donné à la déclaration de la colonne. Par exemple, si une colonne est spécifiée comme SET("a","b","c","d"), alors "a,d", "d,a", & "d,a,d,d" seront tous représentés par "a,d".

OCCUPATION MÉMOIRE DES DONNÉES NON NUMÉRIQUES.

TYPE	ESPACE REQUIS
CHAR(*M*)	M octets, 1 <= M <= 255
VARCHAR(*M*)	M+1 octets, avec 1 <= M <= 255
TINYBLOB, TINYTEXT	M+1 octets, avec M < 2^8
BLOB, TEXT	M+2 octets, avec M < 2^16

TYPE	ESPACE REQUIS
MEDIUMBLOB, MEDIUMTEXT	M+3 octets, avec M < 2^24
LONGBLOB, LONGTEXT	M+4 octets, avec M < 2^32
ENUM('valeur1','valeur2',...)	1 ou 2 octets, suivant le nombre d'éléments de l'énumération (65535 au maximum)
SET('valeur1','valeur2',...)	1, 2, 3, 4 ou 8 octets, suivant le nombre de membres de l'ensemble (64 au maximum)

LES DONNÉES TEMPORELLES

Il y a cinq format de données temporelles en SQL : trois simples un do,,a,t la date, l'autre l'heure ou le temps & le dernier l'année ; & deux horodatages le premier formaté & le second non !

NOM	DESCRIPTION
DATE	Date au format japonais AAAA-MM-JJ.
DATETIME	Date & heure au format japonais AAAA-MM-JJ HH:MM:SS.
TIMESTAMP	Affiche la date & l'heure sans séparateur : AAAAMMJJHHMMSS.
TIMESTAMP(*M*)	Idem mais *M* vaut un entier pair entre 2 & 14. Affiche les *M* premiers caractères de TIMESTAMP.

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

NOM	DESCRIPTION
TIME	Heure au format HH:MM:SS.
YEAR	Année au format AAAA.

Les dates doivent êtres données sous la forme année-mois-jour (exemple : *98-09-21*). La place requise varie d'1 octet (YEAR) à 8 octets (DATETIME) en passant par 3 (DATE ou TIME) & 4 (TIMESTAMP).

Les Commandes : instructions & requêtes

Si les requêtes ont pour but d'extraire des informations d'une base, les commandes vissent à en modifier durablement le contenu.

Elles sont classés en six groupes :

- définition de données (**CREATE DATABASE**, **DROP TABLE**, **ALTER INDEX**, etc.),
- manipulation de données (**INSERT**, **DELETE**, **SELECT**, etc.),
- utilisateur (**DESCRIBE**, **USE**),
- relatives aux verrous & aux transactions (**LOCK TABLE**, **COMMIT**, etc.),
- réplication (**RESET MASTER**, **START SLAVE**, etc.),
- commandes d'administration (**CREATE USER**, **GRANT**, **BACK-UP TABLE**, **REPAIR TABLE**, **SHOW TABLE**, **FLUSH**, etc.)

Nous nous intéresserons un peu aux deux premiers & beaucoup aux deux derniers.

Définition & manipulation des données

Avertissement

Cette section manque d'exemples, car son objet s'avère avant tout culturel. Il faut connaître l'existence de ces commandes & leur syntaxe afin de faciliter le diagnostic en cas de problème. L'administrateur réseaux emploiera les utilitaires de maintenance accompagnant le serveur &, si possible, des client web comme phpMyAdmin, & laissera à l'administrateur de bases de données le privilège d'employer le SQL.

Néanmoins, un effort a été fait pour améliorer la syntaxe des instructions confuse dans les documentations de MySQL & de MariaDB.

Dans un client en mode texte, toutes les instructions doivent se terminer par un point-virgule. Les clients graphiques le rajoutent quand il est manquant.

Ces commandes commencent toutes par un verbe (**CREATE**, **DROP**, **ALTER**, **INSERT**, etc.) ; elles sont, le plus souvent, suivi d'un nom d'objet (**BASE**, **TABLE**, **INDEX**, **ROW**) & de clauses éventuelles.

La gestion des utilisateurs, la sauvegarde & la restauration de tables & de bases de données s'avèrent, également, fondamentales.

Une attention toute particulière sera portée à l'instruction **SELECT**, équivalente au print ou à echo dans d'autres langages, mais, également, avec des expressions SQL pouvant s'avérer complexes.

DÉFINITION DES DONNÉES

Elles sont de deux sortes selon qu'elles permettent de manipuler les bases, les tables ou les lignes.

À chacune de ces instructions correspond un droit qu'il faut avoir pour pouvoir les exécuter. Par exemple, un utilisateur ne peut créer une table que s'il a le droit CREATE sur les tables.

BASES

CREATE DATABASE

Crée une nouvelle base de données vide.

CREATE DATABASE [IF NOT EXISTS] *nom_bdd* [clause_BDD [[,] clause_BDD] ...]

clause_BDD :

Ces caractéristiques sont stockées dans le fichier db.opt du dossier de base.

1 [DEFAULT] CHARACTER SET nom_ens_char | [DEFAULT] COLLATE nom_collation

Exemple 27 :

```
CREATE DATABASE `information_schema` DEFAULT
CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Notez l'usage du caractère ` pour isoler l'identifiant de la base de données. Son emploi est facultatif, mais il est fortement recommandé : le SQL étant un langage extrêmement verbeux (plus de 500 mots initiaux), il vous évitera les effets de bords dus à un nom (de base, de table ou de colonne) identique à un mot du langage.

ALTER DATABASE

Modifie les caractéristiques générales d'une base de données.

```
ALTER DATABASE nom_bdd [clause_BDD [, clause_BDD] ...]
```

DROP DATABASE

Détruit complètement une base de données existante.

```
DROP DATABASE [IF EXISTS] nom_bdd
```

USE

Définit une base de données comme base par défaut.

```
USE nom_bdd
```

TABLES

Elles sont de deux sortes les tables de données où vos données sont effectivement stockées & les tables d'index gérées par *MySQL* qui servent à accélérer l'accès aux données stockées dans les premières.

CREATE TABLE

Crée une nouvelle table.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nom_table
[(définition_create,...)]
[options_table] [instruction_select]
ou
```

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nom_table
[(] LIKE ancien_nom_table [)];
```

définition_create

```
1   définition_colonne
2   | [CONSTRAINT [symbole]] PRIMARY KEY [type_index]
(nom_col_index,...)
3   | KEY [nom_index] [type_index] (nom_col_index,...)
4   | INDEX [nom_index] [type_index] (nom_col_index,...)
5   | [CONSTRAINT [symbole]] UNIQUE [INDEX]
6   [nom_index] [type_index] (nom_col_index,...)
7   | [FULLTEXT|SPATIAL] [INDEX] [nom_index] (nom_col_index,...)
8   | [CONSTRAINT [symbole]] FOREIGN KEY
9   [nom_index] (nom_col_index,...) [définition_référence]
10  | CHECK (expr)
```

définition_colonne

```
1   nom_col type [NOT NULL | NULL] [DEFAULT valeur_défaut]
[AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'chaîne']
[définition_référence]
```

type

```
1   TINYINT[(longueur)] [UNSIGNED] [ZEROFILL]
2   | SMALLINT[(longueur)] [UNSIGNED] [ZEROFILL]
3   | MEDIUMINT[(longueur)] [UNSIGNED] [ZEROFILL]
4   | INT[(longueur)] [UNSIGNED] [ZEROFILL]
5   | INTEGER[(longueur)] [UNSIGNED] [ZEROFILL]
6   | BIGINT[(longueur)] [UNSIGNED] [ZEROFILL]
7   | REAL[(longueur,décimales)] [UNSIGNED] [ZEROFILL]
8   | DOUBLE[(longueur,décimales)] [UNSIGNED] [ZEROFILL]
9   | FLOAT[(longueur,décimales)] [UNSIGNED] [ZEROFILL]
10  | DECIMAL(longueur,décimales) [UNSIGNED] [ZEROFILL]
```

```
11    | NUMERIC(longueur,décimales) [UNSIGNED] [ZEROFILL]
12    | DATE
13    | TIME
14    | TIMESTAMP
15    | DATETIME
16    | CHAR(longueur) [BINARY | ASCII | UNICODE]
17    | VARCHAR(longueur) [BINARY]
18    | TINYBLOB
19    | BLOB
20    | MEDIUMBLOB
21    | LONGBLOB
22    | TINYTEXT
23    | TEXT
24    | MEDIUMTEXT
25    | LONGTEXT
26    | ENUM(valeur1,valeur2,valeur3,...)
27    | SET(valeur1,valeur2,valeur3,...)
28    | type_spatial
```

nom_col_index

```
1     nom_col [(longueur)] [ASC | DESC]
```

définition_référence

```
1     REFERENCES nom_table [(nom_col_index,...)]
2     [MATCH FULL | MATCH PARTIAL]
3     [ON DELETE option_référence]
4     [ON UPDATE option_référence]
```

option_référence

```
1     RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

options_table

```
option_table [option_table] ...
```

option_table

```
{ENGINE|TYPE} = {BDB|HEAP|ISAM|InnoDB|MERGE|MRG_MYISAM|
MYISAM}
  | AUTO_INCREMENT = valeur
  | AVG_ROW_longueur = valeur
  | CHECKSUM = {0 | 1}
  | COMMENT = 'chaîne'
  | MAX_ROWS = valeur
  | MIN_ROWS = valeur
  | PACK_KEYS = {0 | 1 | DEFAULT}
  | PASSWORD = 'chaîne'
  | DELAY_KEY_WRITE = {0 | 1}
  | ROW_FORMAT = { DEFAULT | DYNAMIC | FIXED | COMPRESSED }
  | RAID_TYPE = { 1 | STRIPED | RAID0 }
  | RAID_CHUNKS = valeur
  | RAID_CHUNKSIZE = valeur
  | UNION = (nom_table[,nom_table]...)
  | INSERT_METHOD = { NO | FIRST | LAST }
  | DATA DIRECTORY = 'chemin absolu du dossier'
  | INDEX DIRECTORY = 'chemin absolu du dossier'
  | [DEFAULT] CHARACTER SET nom_ens_car [COLLATE
nom_collation]
```

type spatial

Reportez-vous au chapitre du manuel SQL qui lui est consacré (souvent intitulé données spatiales ou données géométriques).

instruction_select

```
[IGNORE | REPLACE] [AS] SELECT (suite d'une instruction
select licite cf. p. 140) ...
```

Exemple 28 :

```
CREATE TEMPORARY TABLE `CHARACTER_SETS` (
  `CHARACTER_SET_NAME` varchar(32) NOT NULL DEFAULT '',
  `DEFAULT_COLLATE_NAME` varchar(32) NOT NULL DEFAULT '',
  `DESCRIPTION` varchar(60) NOT NULL DEFAULT '',
  `MAXLEN` bigint(3) NOT NULL DEFAULT '0'
) ENGINE=MEMORY DEFAULT CHARSET=utf8;
```

CREATE INDEX

Ajout d'index à une table existante.

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX nom_index
[type_index]
ON nom_table (nom_col_index,...)

nom_col_index
1   nom_col [(longueur)] [ASC | DESC]
```

ALTER TABLE

Modifie la définition d'une table existante.

```
ALTER [IGNORE] TABLE nom_table
clause_alter [, clause_alter] ...

clause_alter
1   ADD [COLUMN] définition_colonne [FIRST | AFTER nom_col ]
2   | ADD [COLUMN] (définition_colonne, ...)
3   | ADD INDEX [nom_index] [type_index] (nom_col_index, ...)
4   | ADD [CONSTRAINT [étiq_contrainte]]
5   PRIMARY KEY [type_index] (nom_col_index, ...)
6   | ADD [CONSTRAINT [étiq_contrainte]]
7   UNIQUE [nom_index] [type_index] (nom_col_index, ...)
8   | ADD [FULLTEXT|SPATIAL] [nom_index] (nom_col_index, ...)
9   | ADD [CONSTRAINT [étiq_contrainte]]
```

```
10    FOREIGN KEY [nom_index] (nom_col_index, ...)
11    [définition_référence]
12    | ALTER [COLUMN] nom_col {SET DEFAULT littéral | DROP
    DEFAULT}
13    | CHANGE [COLUMN] old_nom_col définition_colonne
14    [FIRST|AFTER nom_col]
15    | MODIFY [COLUMN] définition_colonne [FIRST | AFTER
    nom_col]
16    | DROP [COLUMN] nom_col
17    | DROP PRIMARY KEY
18    | DROP INDEX nom_index
19    | DROP FOREIGN KEY étiq_contrainte_fk
20    | DISABLE KEYS
21    | ENABLE KEYS
22    | RENAME [TO] nouveau_nom_table
23    | ORDER BY nom_col
24    | CONVERT TO CHARACTER SET nom_ens_car [COLLATE
    nom_collation]
25    | [DEFAULT] CHARACTER SET nom_ens_car [COLLATE
    nom_collation]
26    | DISCARD TABLESPACE
27    | IMPORT TABLESPACE
28    | options_tables
```

littéral

Une constant chaîne de caractère ou numérique

DROP TABLE

est utilisée pour détruire complètement une table existante.

```
DROP [TEMPORARY] TABLE [IF EXISTS]
nom_table [, nom_table] ... [RESTRICT | CASCADE]
```

DROP INDEX

Supprime l'index nommé nom_de_l_index de la table nom_de_table

DROP INDEX *nom_de_l_index* **ON** *nom_de_table*

DESCRIBE

Montre la structure d'une table.

{DESCRIBE | DESC} *nom_de_table* **[***nom_de_colonne* **|** joker**]**

joker

Ce peut être un des trois caractères jokers (% – remplacement d'un nombre quelconque de caractères, de colonnes, de tables ou de bases, quelconques, selon le contexte –, _ – remplacement d'un seul caractère quelconque, etc. –, NULL – remplacement de tous les caractères, etc.)

LIGNES

INSERT

REPLACE

Remplace (ajoute ou modifie) des données nouvelles ou existantes.

REPLACE [LOW_PRIORITY | DELAYED]
[INTO] *nom_table* **[(***nom_col***,…)]**
VALUES ({*expr* | **DEFAULT**},…),(…),…
ou :
REPLACE [LOW_PRIORITY | DELAYED]
[INTO] *nom_table*
SET *nom_col*={*expr* | **DEFAULT**}, …
ou :
REPLACE [LOW_PRIORITY | DELAYED]

[INTO] *nom_table* [(*nom_col*,...)]
SELECT ...

DELETE

MANIPULATION DES DONNÉES

SELECT

Calcule, lit ou sélectionne des données.

```
SELECT [STRAIGHT_JOIN] [SQL_SMALL_RESULT]
[SQL_BIG_RESULT] [SQL_BUFFER_RESULT] [SQL_CACHE |
SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
[DISTINCT | DISTINCTROW | ALL] sous_sélection [AS alias][, ...]
[INTO {OUTFILE | DUMPFILE} 'nom_fichier' export_options]
[FROM référence_table
[WHERE définition_where]
[GROUP BY définition_grouporder
[HAVING définition_where]
[ORDER BY définition_grouporder]
[LIMIT [début,] lignes]
[PROCEDURE procedure_name(argument_list)]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

```
définition_grouporder
{entier_non_signé | nom_de_colonne | formule} [ASC | DESC], ...
```

```
export_options
```

Valent également comme option d'importation

```
1  [FIELDS
2  [TERMINATED BY '\t']
3  [[OPTIONALLY] ENCLOSED BY '']
4  [ESCAPED BY '\\' ]
5  ]
```

```
6   [LINES
7   [STARTING BY '']
8   [TERMINATED BY '\n']
9   ]
10  [IGNORE nombre LINES]
11  [(nom_col,...)]
```

définition_where

Toute expression booléenne. Les opérandes de l'expression peuvent être toutes les expressions-valeurs typées (cf. p. 168) ou non typée (cf. p. 166).

définition_grouporder

{entier_non_signé | nom_de_colonne | formule} [ASC | DESC], ...

formule

Une formule est en fait une expression typée.

Jointures

Les jointures sont des tables temporaires résultant de la combinaison de plusieurs tables ou de tables & de jointures.

```
1   référence_table, référence_table
2   référence_table [CROSS] JOIN référence_table
3   référence_table INNER JOIN référence_table condition_jointure
4   référence_table STRAIGHT_JOIN référence_table
5   référence_table LEFT [OUTER] JOIN référence_table
    condition_jointure
6   référence_table LEFT [OUTER] JOIN référence_table
7   référence_table NATURAL [LEFT [OUTER]] JOIN
    référence_table
8   { OJ référence_table LEFT OUTER JOIN référence_table ON
    expr_conditionnelle }
9   référence_table RIGHT [OUTER] JOIN référence_table
```

condition_jointure

10 référence_table RIGHT [OUTER] JOIN référence_table

11 référence_table NATURAL [RIGHT [OUTER]] JOIN référence_table

référence_table

1 *nom_de_table* [[AS] *alias*] [USE INDEX (*clé*[, ...])] [IGNORE INDEX (*clé*[, ...])]

2 référence_table :

3 *nom_de_table* [[AS] *alias*] [USE INDEX (*clé*[, ...])] [IGNORE INDEX (*clé*[, ...])]

condition_jointure

4 ON expr_conditionnelle | USING (*nom_col*[, ...])

expr_conditionnelle

Idem définition_where page précédente.

SOUS-REQUÊTE OU SOUS-SÉLECTION :

Une sous-requête est une commande **SELECT** dans une autre commande. Elle est entourée de parenthèses.

Une sous-sélection peut être utilisée avec les commandes suivantes, **SELECT**, **INSERT**, **UPDATE**, **DELETE**, **SET** & **DO**.

Une sous-sélection peut contenir les mêmes mots-clé & les mêmes clauses qu'une commande **SELECT** (**DISTINCT**, **WHERE**, **GROUP BY**, **ORDER BY**, **LIMIT**, jointures, **UNION**, commentaires, fonctions, etc.)

L'utilisation la plus répandue des sous-sélections est celle-ci :

opérande_non_sous-requête opérateur_de_comparaison (sous-sélection)

où opérateur_de_comparaison est l'un des opérateurs suivants, =, >, <, >=, <=, <>.

Quatre mots clés supplémentaires, qui doivent suivre immédiatement un opérateur de comparaison, peuvent servir d'opérateurs dans ces expressions :

- opérande opérateur_de_comparaison ANY (sous-sélection) ;
- opérande **IN** (sous-sélection) ;
- opérande opérateur_de_comparaison SOME (sous-sélection) ;
- opérande opérateur_de_comparaison ALL (sous-sélection).

Le mot ANY signifie : retourne TRUE si la comparaison est vraie pour UNE des lignes de la sous-requête. IN est synonyme de = ANY & SOME est un alias de ANY. ALL exige que la comparaison soit vraie pour toutes les lignes de la sous-requête.

UNION

Combine les résultat de plusieurs requêtes **SELECT** en un seul résultat. Les colonnes listées dans la partie sous_sélection du **SELECT** doivent être du même type. Les noms de colonnes utilisés dans le premier **SELECT** seront utilisé comme nom de champs pour les résultats retournés.

SELECT expression_select UNION [ALL | DISTINCT] **SELECT** expression_select [UNION [ALL | DISTINCT]] ...

Si vous n'utilisez pas le mot clef **ALL** pour l'UNION, toutes les lignes retournées seront uniques, comme si vous aviez fait un **DISTINCT** pour l'ensemble du résultat. Si vous spécifiez **ALL**, vous aurez alors tout les résultats retournés par toutes les commandes **SELECT**.

On désigne par expression_select, le résultat d'une instruction SELECT ou d'une sous-sélection.

INSERT

Ajoute de nouvelles données.

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]

[INTO] *nom_table* [(*nom_col*,…)]
VALUES ({*expr* | DEFAULT}, …),(…), …
[ON DUPLICATE KEY UPDATE *nom_col*=*expr*, …]
ou
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] *nom_table*
SET *nom_col*={*expr* | DEFAULT}, …
[ON DUPLICATE KEY UPDATE *nom_col*=*expr*, …]
ou
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] *nom_table* [(*nom_col*,…)]
SELECT expression_select …

UPDATE

Modifie les données existantes.

UPDATE [LOW_PRIORITY] [IGNORE] nom_table
SET *nom_col1*=*expr1* [, *nom_col2*=*expr2* …]
[WHERE définition_where]
[ORDER BY définition_grouporder]
[LIMIT *nb_lignes*]

Syntaxe multi-tables :

UPDATE [LOW_PRIORITY] [IGNORE] *nom_table* [, *nom_table* …]
SET *nom_col1*=*expr1* [, *nom_col2*=*expr2* …]
[WHERE définition_where]

DELETE

Supprime des données existantes.

DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM
table_name
[WHERE définition_where]
[ORDER BY définition_grouporder]

[LIMIT *nb_lignes*]
Syntaxe multi-**tables** :
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] *nom_table*[.*] [, *nom_table*[.*] ...]
FROM *nom_table*-references
[WHERE définition_where]
ou :
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
FROM *nom_table*[.*] [, *nom_table*[.*] ...]
USING références_tables
[WHERE définition_where]

TRUNCATE

Équivaut à COMMIT ; DELETE FROM nom_de_table. Conséquences :

- emploi la même syntaxe que DELETE (cf. p. 154) ;
- dangereux, car ignore les transactions en cours.

TRUNCATE TABLE *nom_de_table*

TRANSACTIONS

Ces instructions sont mentionnées pour votre culture, car, par défaut, MySQL est lancé en mode ***autocommit***. Cela signifie que chaque modification effectué est enregistré immédiatement sur le disque.

Si vous utilisez des tables supportant les transactions (comme InnoDB, BDB), vous pouvez configurer MySQL en mode ***non-autocommit*** grâce à la commande **SET *AUTOCOMMIT*=0**.

A partir de là, vous devrez utiliser COMMIT pour enregistrer les modifications sur le disque ou ROLLBACK pour ignorer les modifications apportées depuis le début de la transaction.

Si vous souhaitez sortir du mode ***AUTOCOMMIT*** pour une série d'opérations, vous utiliserez la commande START TRANSACTION.

Exemple 29 : Tiré du manuel MySQL 5.0

```
START TRANSACTION;
SELECT @A=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summmary=@A WHERE type=1;
COMMIT;
```

START TRANSACTION

Commence une transaction.

COMMIT

Applique les modifications & termine la transaction.

ROLLBACK

Annule les modifications & termine la transaction.

RÉPLICATION & ADMINISTRATION

Il n'est pas besoin d'une grosse configuration pour avoir besoin de répliquer un serveur : un cyber-café, un petit serveur de jeu en ligne en ont besoin : c'est un des éléments permettant d'assurer la continuité de service.

RÉPLICATION

Elle s'avère une des trois différences fondamentales, avec la gestion des cluster & la gestion de moteurs de base de données NoSQL entre MySQL & MariaDB. Le premier ne connaît que la réplication asymétrique synchrone : quand une transaction a lieur sur le premier serveur dit maître, elle est recopié immédiatement sur le second serveur dit esclave. MariaDB connaît également la réplication symétrique dans la quelle une transaction effectué sur le second serveur est recopiée vers le premier. Ce sont ces différences qui expliquent la migration de Google & Wikipédia vers MariaDB.

Nous n'aborderons que le migration asymétrique, la plus répandue dans les petites structures.

PURGE MASTER LOGS

Efface tous les logs binaires listés dans l'index de logs, qui sont antérieurs à la date ou à l'évènement indiqué. Celui donné en paramètre devient le premier de la liste.

```
PURGE {MASTER | BINARY} LOGS TO 'nom_log'
PURGE {MASTER | BINARY} LOGS BEFORE 'date'
```

Exemple 30 :

```
PURGE MASTER LOGS TO 'mysql-bin.010';
PURGE MASTER LOGS BEFORE '2003-04-02 22:46:26';
```

RESET MASTER

Efface tous les fichiers de logs binaires dans le fichier d'index, & vide le fichier d'index des logs.

```
RESET MASTER
```

SHOCHANGE MASTER TO DÉFINITION_MAÎTRE [, DÉFINITION_MAÎTRE] ...W ...

```
SHOW BINLOG EVENTS [ IN 'nom_log' ] [ FROM prem_ligne ] [
LIMIT [limite,] nb_lignes ]
```

Affiche les événements du log binaire. Si vous ne spécifiez pas 'nom_log', le premier log binaire sera affiché.

```
SHOW MASTER LOGS
```

Liste les logs binaires disponibles sur le maître. Vous devriez utiliser cette commande avant PURGE MASTER LOGS pour savoir jusqu'où vous pouvez aller.

```
SHOW MASTER STATUS
```

Affiche les informations d'état du log binaire du maître.

```
SHOW SLAVE HOSTS
```

Affiche la liste des esclaves actuellement enregistrée sur le maître.

CHANGE MASTER TO

Modifie les paramètres que l'esclave utilise pour se connecter & pour communiquer avec le serveur maître.

```
CHANGE MASTER TO définition_maître [, définition_maître] ...
```

```
définition_maître
MASTER_HOST = 'nom_hôte'
| MASTER_USER = 'nom_utilisateur'
| MASTER_PASSWORD = 'password'
| MASTER_PORT = num_port
| MASTER_CONNECT_RETRY = compte
| MASTER_LOG_FILE = 'nom_log_maître'
| MASTER_LOG_POS = pos_log_maître
| RELAY_LOG_FILE = 'nom_log_relai'
| RELAY_LOG_POS = pos_log_relay
| MASTER_SSL = {0|1}
| MASTER_SSL_CA = 'nom_fic_ca'
| MASTER_SSL_CAPATH = 'nom_dossier_ca'
| MASTER_SSL_CERT = 'nom_fic_cert'
| MASTER_SSL_KEY = 'nom_fic_clé'
| MASTER_SSL_CIPHER = 'liste_cod'
```

Exemple 31 :

```
mysql> STOP SLAVE;  -- Si la réplication est active !
mysql> CHANGE MASTER TO
→ MASTER_HOST='master2.mycompany.com',
→ MASTER_USER='replication',
→ MASTER_PASSWORD='bigs3cret',
→ MASTER_PORT=3306,
```

```
→ MAS LOAD TABLE nom_table FROM
MASTERTER_LOG_FILE='master2-bin.001',
→ MASTER_LOG_POS=4,
→ MASTER_CONNECT_RETRY=10;
mysql> CHANGE MASTER TO
→ RELAY_LOG_FILE='slave-relay-bin.006',
→ RELAY_LOG_POS=4025;
mysql> START SLAVE; -- Si vous voulez redémarrer la réplication !
```

LOAD DATA FROM MASTER

Fait une sauvegarde du maître & la copie vers l'esclave. Met à jour les valeurs de MASTER_LOG_FILE & MASTER_LOG_POS pour que la réplication reprennent à la bonne position. Respecte les interdictions de réplications de tables & de bases.

ADMINISTRATION

GESTION DES UTILISATEURS

La gestion des droits des utilisateurs des SGBD est plus fine que celle traditionnelle des Unix : elle comporte vingt-huit droits se répartissant en droits sur les données, sur la structure ou sur l'administration de la base de donnée. Ces droits peuvent être valables pour toutes les bases de données gérées par un serveur ou limités à une ou plusieurs d'entre elles.

En principe, vous ne devriez pas avoir à modifier les droits ceux-ci étant défini par la catégori à laquelle appartient l'utilisateur (par exemple : client, agent, administrateur).

DONNÉES	STRUCTURE	ADMINISTRATION
SELECT	CREATE	GRANT
INSERT	ALTER	SUPER
UPDATE	INDEX	PROCESS

DONNÉES	STRUCTURE	ADMINISTRATION
DELETE	DROP	RELOAD
FILE	CREATE TEMPORARY TABLES	SHUTDOWN
	SHOW VIEW	SHOW DATABASES
	CREATE ROUTINE	LOCK TABLES
	ALTER ROUTINE	REFERENCES
	EXECUTE	REPLICATION CLIENT
	CREATE VIEW	REPLICATION SLAVE
	EVENT	CREATE USER
	TRIGGER	

CREATE USER, DROP USER & RENAME

```
CREATE USER utilisateur [IDENTIFIED BY [PASSWORD]
'mot_de_passe'] [, utilisateu [IDENTIFIED BY [PASSWORD]
'mot_de_passe']] ...
DROP USER utilisateur
RENAME USER anc_utilisateur TO nouv_utilisateur [, anc_utilisateu
TO nouv_utilisateur] ...
```

GRANT & REVOKE

```
GRANT définition_privilège [(liste_colonne)] [, définition_privilège
[(liste_colonne)]] ...
ON {nom_table | * | *.* | nom_bdd.*}
TO utilisateur [IDENTIFIED BY [PASSWORD] 'mot_de_passe'] [,
utilisateur [IDENTIFIED BY [PASSWORD] 'mot_de_passe']] ...
[REQUIRE
  NONE | [{SSL | X509}]
  [CIPHER code [AND]]
  [ISSUER émetteur [AND]]
  [SUBJECT sujet]]
```

[WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR compte | MAX_UPDATES_PER_HOUR compte | MAX_CONNECTIONS_PER_HOUR compte]]
REVOKE définition_privilège [(liste_colonne)] [, définition_privilège [(liste_colonne)]] …
ON {nom_table | * | *.* | nom_bdd.*}
FROM utilisateur [, utilisateur] …
REVOKE ALL PRIVILEGES, GRANT OPTION FROM utilisateur [, utilisateur] …

On le voit les droits peuvent être :

- *au niveau global*, s'appliquant à toutes les bases de données d'un serveur ; stockés dans la table mysql.user, ils sont supprimés par **REVOKE ALL ON** *.* ;
- *au niveau d'une base de données*, s'appliqueat à toutes les tables d'une base de données ; stockés dans les tables mysql.db & mysql.host ; ils sont supprimés par **REVOKE ALL ON** db.* ;
- *au niveau d'une table*, s'appliquant à toutes les colonnes d'une table ; stockés dans la table mysql.tables_priv, ils sont supprimés par **REVOKE ALL ON** db.table ;
- *au niveau d'ils sont supprimés par une colonne ou plusieurs colonnes*, s'appliquant à une colonne d'une table ; stockés dans la table mysql.columns_priv, ils sont supprimés par **REVOKE ALL ON** bdd.table.colonne ….

GESTION DE L'ENTRETIEN DES TABLES

Bien que l'on puisse le réaliser à partir des logiciels associés à MariaDB, il peut s'avérer plus commode de le faire avec un des client MySQL & les instructions suivantes.

{ANALYZE|OPTIMIZE|BACKUP|REPAIR|CHECKSUM|RESTORE} TABLE

Ces commandes, mentionnées pour mémoire, ne fonctionnent qu'avec des tables MyISAM, le moteur de base de données originel de *MySQL*, délaissé au profit d'InnoDB qui gère les transactions. C'est la raison pour laquelle l'usage d'utilitaires gérant les tables des autres moteurs de base de données s'avère indispensable.

CHECK TABLE

Vérifie l'intégrité des tables.

```
CHECK TABLE nom_table[,nom_table...] [option [option...]]

option = QUICK | FAST | MEDIUM | EXTENDED | CHANGED
```

SET

Configure plusieurs options qui affectent le comportement de votre serveur ou de votre client.

```
SET assignation_var [, assignation_var] ...

assignation_var :
  @nom_var_util = expr
  | [GLOBAL|S@ESSION] nom_var_syst = expr
  | @@[global.|session.]nom_var_syst = expr
```

Les variables système peuvent être identifiées dans une commande **SET** sous la forme *nom_var*. Le nom peut être optionnellement précédé par GLOBAL ou *@@global.* pour indiquer que cette variable est globale, ou par SESSION, *@@session.*, ou *@@* pour indiquer que cette variable est une variable de session. LOCAL & *@@local.* sont synonymes de SESSION & *@@session.*. Si aucune option n'est présente, **SET** spécifie une variable de session.

La syntaxe *@@var_name* pour les variables système est supportée pour rendre la syntaxe *MySQL* compatible avec les autres SGBD.

Si vous configurez plusieurs variables sur une seule ligne de commande, le dernier mode GLOBAL | SESSION utilisé est pris en compte.

Exemple 32 :

```
SET sort_buffer_size=10000;
SET @@local.sort_buffer_size=10000;
SET GLOBAL sort_buffer_size=1000000, SESSION
sort_buffer_size=1000000;
SET @@sort_buffer_size=1000000;
SET @@global.sort_buffer_size=1000000,
@@local.sort_buffer_size=1000000;
```

Si vous utilisez SESSION (par défaut) l'option que vous configurez garde son effet jusqu'à ce que la session courante se termine, ou que vous modifiez à nouveau cette option. Si vous utilisez GLOBAL, qui requière le privilège SUPER, l'option est gardée en mémoire & utilisée pour les nouvelles connexions jusqu'au redémarrage du serveur. Si vous voulez qu'un changement reste permanent, vous devez l'effectuer dans l'un des fichiers d'options de *MySQL*.

Pour éviter un mauvais usage, *MySQL* donnera une erreur si vous utilisez **SET** GLOBAL avec une variable qui ne peut être inutilisée que par **SET** SESSION ou si vous n'utilisez pas **SET** GLOBAL avec une variable globale.

Si vous voulez configurer une variable SESSION à une valeur GLOBAL ou une valeur GLOBAL à la valeur par défaut de *MySQL*, vous pouvez la configurer à DEFAULT.

Exemple 33 :

```
SET max_join_size=DEFAULT;
SET @@session.max_join_size=@@global.max_join_size;
```

SHOW VARIABLES donne la liste de la plupart des variables globales. Vous pouvez obtenir la valeur d'une variable spécifique avec la syntaxe *@@[global.|local.]nom_variable* :

Exemple 34 :

```
SHOW VARIABLES LIKE 'max_join_size';
SHOW GLOBAL VARIABLES LIKE 'max_join_size';
```

Vous pouvez aussi obtenir une valeur spécifique d'une variable en utilisant la syntaxe *@@[global.|local.]var_name* avec **SELECT** :

Exemple 35 :

```
SELECT @@max_join_size, @@global.max_join_size;
```

Lorsque vous lisez la valeur d'une variable avec la syntaxe **SELECT** *@@var_name* (c'est à dire, sans spécifier global., session. ou local.), *MySQL* retourne la valeur de SESSION si elle existe, & la valeur GLOBAL sinon.

AFFICHAGE D'INFORMATIONS

SHOW ...

Donne des informations sur les bases, tables, colonnes ou sur le serveur.

```
SHOW
 [FULL] COLUMNS FROM nom_table [FROM nom_bdd] [LIKE
'modèle']
| CREATE DATABASE nom_bdd
| CREATE TABLE nom_table
| DATABASES [LIKE 'modèle']
| [STORAGE] ENGINES
| ERRORS [LIMIT [début,] nb_lignes]
| GRANTS FOR user
| INDEX FROM nom_table [FROM nom_bdd]
| INNODB STATUS
```

```
| [BDB] LOGS
| PRIVILEGES
| [FULL] PROCESSLIST
| STATUS [LIKE 'modèle']
| TABLE STATUS [FROM nom_bdd] [LIKE 'modèle']
| [OPEN] TABLES [FROM nom_bdd] [LIKE 'modèle']
| [GLOBAL | SESSION] VARIABLES [LIKE 'modèle']
| WARNINGS [LIMIT [début,] nb_lignes]
```

Exemple 36 :

LES EXPRESSIONS

La syntaxe de beaucoup d'instructions comporte le mot expression ou un terme équivalent spécialisé comme SousRequêteTable. Les expression sont autorisées à des emplacements précis des instructions.

Certaines instructions n'acceptent que des expressions particulières ou possédant un propriété précise. Sans spécification particulières, elles sont autorisées partout ou le mot expression figure dans la syntaxe. Elles sont de cinq sortes :

- clauses ORDER BY,
- expressions SELECT,
- parties SET d'une instruction UPDATE,
- expressions valeurs,
- clauses WHERE.

Bien sûr, plusieurs autres instructions incluent ces éléments dans leur construction.

Les tables suivantes listent toutes les expressions SQL possibles & indiquent où elles sont autorisées.

EXPRESSIONS VALEURS

GÉNÉRALES

Les expressions générales sont des expressions dont le résultat peut être une valeur de n'importe quel type.

NOM	EXPLICATION
Référence de colonne Column reference	Un NomDeColonne donne accès aux valeurs qu'il référence dans l'expression le contenant. Dans **SELECT** *Nom* **FROM** *Realisateur*, *Nom* donne accès aux noms de tous les réalisateurs listés dans la table *Rea-*

NOM	EXPLICATION
	lisateur. Cette référence est qualifiée par le nom de la table la contenant, ou éventuellement par son alias, s'il y a risque d'ambiguïté. S'il existe un alias, il doit obligatoirement être employé à la place du nom de la table.
	Expressions SELECT, Instructions UPDATE, clause WHERE
Constante Constant	La plupart des types de données ont des constantes associée (exemple d'exception : VARCHAR)
NULL	NULL est une constante sans type représentant une valeur inconnue.
	Expressions CAST, listes INSERT valeurS, clauses UPDATE SET. Avec une expression cast, cela lui donne un le type indiqué valeurS CAST (NULL AS BOOLEAN)
Paramètre dynamique Dynamic parameter Bind parameter	C'est un paramètre d'une instruction SQL dont la valeur n'est pas spécifiée. À la place de la valeur on indique ?, :nomparam, @nomparam. Il faut indiquer leur valeur lors de l'exécution. Les valeurs indiquées doivent du type attendu. PREPARE EXPARDYN1 FROM 'SELECT Nom, Prenom FROM Employes WHERE CEmploye = :code' ; EXECUTE EXPARDYN1 USING 10 ; +-------+--------+ | Nom | Prenom | +-------+--------+ | SCIFO | Michel | +-------+--------+ DEALLOCATE EXPARDYN1 ;
	Ils ne sont autorisés que dans les instructions préparées.

NOM	EXPLICATION
Expression CAST	Elles permettent de spécifier un type pour une constante NULL, le type d'un paramètre dynamique ou de modifier le type d'une valeur.
Sous-requêtes (Une sous-requête est une instruction select entre parenthèses)	
scalaire Scalar subquery	Sous-requête fournissant une valeur scalaire (intersection d'une ligne & d'une colonne) d'un des types scalaires (les BLOB sont exclus) Si T1 ne contient qu'une colonne & qu'une ligne : SELECT (SELECT N1 FROM T1) FROM T2 ;
	Seules, les instructions nécessitant une valeur littérale, comme LIMIT, en interdisent l'usage.
Ligne Row subquery	Sous-requête retournant plusieurs colonnes d'une seule ligne ou de plusieurs.
	Expressions ligne & table, clauses FROM, opérateurs EXISTS, IN, ALL, ANY, SOME.
table Table subquery	Sous-requête retournant plusieurs colonnes ou plusieurs lignes.
	Expression table, clauses FROM, opérateurs EXISTS, IN, ALL, ANY, SOME.
Expression select	Un des trois types de sous-requête

LES EXPRESSIONS VALEURS TYPÉES

BOOLÉENNES

Expressions dont les opérandes ou le résultat sont des booléens. Il y a trois valeurs possibles TRUE (1), FALSE (0) & NULL (inconnue). Les opérateurs booléens sont NOT (!), AND (ou &&), OR (ou ||), XOR. Les comparaisons fournissent également un résultat booléen.

NUMÉRIQUES

Elles ont pour résultat une valeur numérique qui a un des types suivant : BIGINT, DECIMAL, DOUBLE PRECISION, INTEGER, REAL, SMALLINT

Outre les opérateurs numériques déjà présentés, trois fonctions sont très utiles :

- AVG, calcule la moyenne ;
- SUM, calcule la somme ;
- COUNT compte les valeurs.

CARACTÈRES

Elles ont pour résultat une valeur CHAR ou VARCHAR.

TYPE	EXPLICATIONS
valeurs CHAR ou VARCHAR avec jokers.	% remplace 0 ou plus caractères _ remplace exactement 1 caractère.
	Opérateur LIKE
Expression concaténation	Dans une telle expression, l'opérateur de concaténation, ‖, accole l'opérande de droite à la fin de l'opérande de gauche. S'applique aux chaînes de caractères ou d'octets.
Fonctions chaînes incorporées	S'appliquent aux chaînes de caractères. cf. LTRIM, LCASE, LOWER, RTRIM, TRIM, SUBSTR, UCASE, UPPER, etc.
Fonctions USER	Fonctionnant retournant une chaîne informant sur l'utilisateur, cf. CURRENT_USER, SESSION_USER, etc.

TEMPORELLES

Elles ont pour résultat une valeur date, temps ou horodatage.

FONCTION	ACTION
CURRENT_DATE	Retourne la date courante
CURRENT_TIME	Retourne l'heure courante
CURRENT_TIMESTAMP	Retourne l'horodatage courant

Exercices

Manipulation des données

Ex. 01

Ex. 02

Ex. 03

Ex. 04

Administration

Ex. 11

Ex. 12

Notes

01001

Nous expliquerons en détail le fonctionnement de ces logiciels. *CSS* ou *HTML* désignent le langage, *css* ou *html* les fichiers ou les mots de ces langages. La notation *(X)HTML* est usuelle pour indiquer que la page peut écrite en *HTML* ou dans une extension du *HTML* le *XHTML*. Les parenthèses proviennent de l'existence d'un flou dans la définition des extensions, en attendant la généralisation du *XML*.

01002

C'est à cela que servent les cookies, ces petits fichiers textes stockés sur votre ordinateur.

Si vous n'aimez être fiché, supprimez tous les cookies régulièrement, à la fin de votre session internet, par exemple, ou au minimum une fois par semaine.

01003

Rappel, les URI sont de deux sortes les URN qui permettent de caractériser un objet sans dire où il se trouve (comme les numéros ISBN pour les livres ou ISSN pour les revues) & les URL qui permettent de le trouver, les URL ne sont pas spécifiques au protocoles HTPP en voici quelques exemples :

- ftp://www.scifo.fr/rfc/rfc1808.txt : protocole FTP,
- http://www.scifo.fr/wp_content/debut.php : protocole HTTP,
- mailto:michel.scifo@afpa.fr : protocole mailto,
- ssh://utilisateur@example.fr : protocole SSH,
- file:///chemin/absolu/vers/une/ressource :fichiers locaux,
- & quelques chemins relatifs :
  - ../../../ressource,

- ./ressource#fragment,
- ressource,
- #fragment.

01004

Les attributs sont les caractéristiques d'une balise. Ainsi, la balise <p>, déjà vue, avait un attribut **align** valant right, left, center ou justified. Cet attribut n'existe plus en HTML 5, car l'alignement est géré dans un fichier *css*.

01005

Si nous devions récrire cette ligne, nous mettrions son contenu dans une feuille de style *css* & nous ne mentionnerions que le nom du style ainsi défini dans la page HTML.

02001

Les mots *position*, *color*, *top*, *left*, *width*, *height* désignent des propriétés de l'attribut **style**. Ils ne devraient plus apparaître dans les pages html uniquement dans les *css*.

02002

On parle de L4G ou langage de 4e génération, c'est-à-dire, d'un langage permettant de développer des applications en minimisant le travail de programmation. *Access* ou *4eDimension* sont des langages de quatrième génération, tout comme les langages de macro-commandes intégrés dans *MSWord*, *MSExcel*, *OpenOffice.org*[*LibreOffice*] *Base*, *Calc* ou *Writer*.

02003

Dans un SGBD relationnel, le mot *relation* est synonyme de *table*, parce qu'une table met en relation les informations caractéristiques d'un objet. Cependant, il est souvent employé pour désigner les liens entre deux tables qui représentent en fait, une opération sur les relations-tables appe-

lée jointure (jargon) ou jonction (français). Une jointure est dite interne si le lien n'existe que quand une valeur est commune à une colonne de chacune des tables-relations concernées. Elle est dite externe, quand le lien intègre les lignes d'une des tables n'ayant pas de données communes. Le problème est que l'on emploie aussi ces mots pour désigner des relations permanentes permettant de résoudre les problèmes d'intégrité référentielle, grâce aux clés permanentes, appelées principale & étrangère, ou des relations ponctuelles établies par les clauses JOIN ou WHERE de l'instruction SELECT. Pire, on emploie également ces termes pour désigner comme interne des relations permanentes ou ponctuelles dans la base de données & comme externe des hyperliens.

02004

Dans les très grandes entreprises, il s'avère indispensable d'analyser finement les données produites par le système d'information, mais celles-ci ont un gros défaut : elles sont variables. Afin d'éviter cette inconvénient on les stocke dans un entrepôt, comme on emmagasinerait, chaque jour la production de la veille, en attendant sa vente. Ces entrepôts sont donc des ordinateurs conservant la production quotidienne d'information de l'entreprise sans la modifier, à des fins d'analyses. Ces dernières n'ont pas les mêmes contraintes que la production, c'est pourquoi elles nécessitent une organisation différente des données On appelle dimension la donnée centrale d'une analyse (on parlera de dimension vente, de dimension stock, etc.).

02005

Le moteur de recherche de Google fonctionne avec un système d'exploitation *Linux* & un SGBD *MySQL/MariaDB* adaptés à ses propres besoins & des logiciels développés en interne, dont le serveur web. De même, ses serveurs sont fabriqués en interne. Les solutions retenues semblent toujours artisanales, mais elles sont avant tout efficaces & économiques. Le

défaut majeur de Google est politique : son quasi-monopole est aussi malsain que celui de Microsoft ou, dans un autre domaine stratégique celui de Monsanto.

02006

Une transaction transfère effectivement une modification de la base de données entre le client & le serveur. Sa gestion varie énormément d'un SGBD à l'autre.

02007

Une transaction sur une *base acide* c'est un oxymore. C'est de l'humour d'informaticien !

03001

Avec Linux, l'activation de xampp, se fait en mode texte, à l'aide de la commande *lampp* installée dans le dossier /opt/lampp.

```
$ /opt/lampp/lampp
Usage: lampp <action>
        start           Start XAMPP (Apache, MySQL and eventually
others)
        startapache     Start only Apache
        startmysql      Start only MySQL
        startftp        Start only ProFTPD
        stop            Stop XAMPP (Apache, MySQL and eventually others)
        stopapache      Stop only Apache
        stopmysql       Stop only MySQL
        stopftp         Stop only ProFTPD
        reload          Reload XAMPP (Apache, MySQL and eventually
others)
        reloadapache    Reload only Apache
        reloadmysql     Reload only MySQL
        reloadftp       Reload only ProFTPD
```

restart Stop and start XAMPP
security Check XAMPP's security
enablessl Enable SSL support for Apache
disablessl Disable SSL support for Apache
backup Make backup file of your XAMPP config, log and data files
oci8 Enable the oci8 extenssion
panel Starts graphical XAMPP control panel

Le panneau de commande graphique sera disponible un jour en saisissant la commande **/opt/lampp/lampp** panel. En attendant on peut l'activer par **/opt/lampp/manager-linux-x64.run**, mais comme il est bien moins complet que celui de Windows, la ligne de command ereste indispensable.

03002

Avec Linux, il faudra saisir **./lampp** security dans le dossier /opt/lampp, puis répondre aux questions.

04001

C'est le sens premier de l'expression, dans lequel le processeur est virtuel. Avec le développement de la simulation logicielle, on parle maintenant de machines virtuelles pour des logiciels simulant, avec un même processeur, un autre système d'exploitation.

04002

Grammaire : ensemble de règles conventionnelles qui déterminent le bon usage d'une langue.

Syntaxe : partie de la grammaire traditionnelle qui étudie les relations entre les mots constituant une proposition ou une phrase, leurs combinaisons, & les règles qui président à ces relations, à ces combinaisons.

04003

Il ne faut pas confondre la *conversion* avec le *transtypage* : le mécanisme de *conversion* transforme effectivement la donnée (par exemple, conversion d'un entier, codé en complément à 2, en réel, codé selon une norme I3E –possible dans tous les langages), en revanche, le *transtypage* se contente de modifier le type associé (possible uniquement avec des langages orientés objets). Ce mécanisme peut être explicite ou implicite.

04004

Les langages de script intégrés à un logiciel bureautique sont dénommés langages de macro-commandes.

05001

Trois acronymes voisins existent : LAMP, WAMP, MAMP. Dans les trois cas, les dernières lettres signifient : *Apache* (serveur web), *MySQL* (serveur de bases de données) & *PHP* (langage de scripts) ; la première lettre indique le système d'exploitation : Linux, Windows, MacOS. Certains ajoutent un second langage de scripts *Perl* & donc un second P (LAMPP, WAMPP). Le X de *XAMPP* indique qu'il fonctionne avec Windows aussi bien qu'avec Linux. Il s'agit des plate-formes de base pour les applications dites Web 2.0.

i Cette valeur a été lue sur un serveur Linux, sur un serveur Windows, les variables d'environnement ont des libellés différents.

ii Certains parlent, également, de *sciles*, mais c'est dû à un contre-sens : si l'acronyme anglais *FIFO* symboleise bien les files (*First In-First Out*), *SCIFO* ne signifie pas *SeCond In-First Out*).

iii Linuxiens attention : c'est le contraire du *Bash* !

iv À mon sens, ce type n'a plus de raisons d'être autres qu'historiques (compatibilité avec des applications existantes).

v Ces types n'ont pas grand sens, puisque les VARCHAR & les VARBINARY sont d'un emploi plus aisés.